

Part IV

Building Safe and Beneficial AI Agents

The rapid development of LLM-based agents introduces a new set of safety challenges that go beyond those of traditional LLMs. Equipped with advanced reasoning, planning, and tool-using capabilities, these agents are designed to perform tasks autonomously and interact with their environments [34]. However, this autonomy also expands the attack surface, creating new vulnerabilities that demand careful research and attention [1129, 40]. In this part, we first establish a comprehensive framework for understanding agent safety, examining both internal and external safety threats to AI agents. We will explore the various attack vectors associated with these threats and propose potential mitigation strategies. This framework is organized into two key areas:

(1) Intrinsic Safety threats stem from vulnerabilities in the agent’s core components, which include the LLM “brain” as well as the perception and action modules. Each of these components has unique weaknesses that can be exploited by adversaries:

- *Brain* is the LLM itself, responsible for key decision-making tasks such as reasoning and planning. It is guided by a knowledge module that provides essential contextual information.
- *Perception* consists of sensors that interpret the external environment, where malicious manipulation of external objects can lead to erroneous perceptions.
- *Action* is responsible for tool usage and downstream applications, which are also susceptible to exploitation.

(2) Extrinsic Safety threats arise from interactions between the agent and external, often untrusted, entities. These include:

- *Agent-Memory Interactions*: The agent frequently accesses and interacts with memory storage, which serves as an external database for decision-making and contextual information retrieval. Recent research highlights vulnerabilities in the agent-memory interface that could be exploited to manipulate the agent’s actions.
- *Agent-Agent and Agent-Environment Interactions*: These refer to the interactions between the agent and other agents (e.g., other agents or human operators), as well as its environment, which includes task-related objects or dynamic systems. The complexity of these interactions further compounds the agent’s exposure to external threats.

As illustrated in Figure 17.1, these risks are broadly categorized into intrinsic and extrinsic safety, helping to clarify their origin and nature. In addition to identifying threats, we also provide a rigorous, mathematical foundation for understanding attacks such as jailbreaking, prompt injection, and data poisoning. Moreover, we present practical, actionable solutions, tracing the development of safety measures from early LLM safeguards to comprehensive strategies that protect the entire agent system. This includes exploring guardrails, advanced alignment techniques (such as superalignment), and the crucial balance between safety and helpfulness. Finally, we analyze the “scaling law of AI safety”—the complex relationship between an agent’s capabilities and its potential risks—and the essential trade-offs that must be made. This part provides a clear understanding of the challenges, theoretical foundations, and practical strategies necessary to develop effective and trustworthy AI agents that can be safely and effectively deployed in real-world scenarios.

This part is organized as follows: First, we examine intrinsic safety risks (Chapter 18), focusing on threats to the LLM “brain,” as well as vulnerabilities in the agent’s perception and action components (Chapter 19). Next, we explore extrinsic safety threats related to agent-memory, agent-agent, and agent-environment interactions (Chapter 20). Finally, we investigate superalignment techniques aimed at ensuring the safety of agent behaviors, while addressing the broader challenge of balancing safety with performance. This includes exploring how safety measures scale with the increasing capabilities of AI systems and examining the trade-offs involved in designing secure, capable AI agents (Chapter 21).

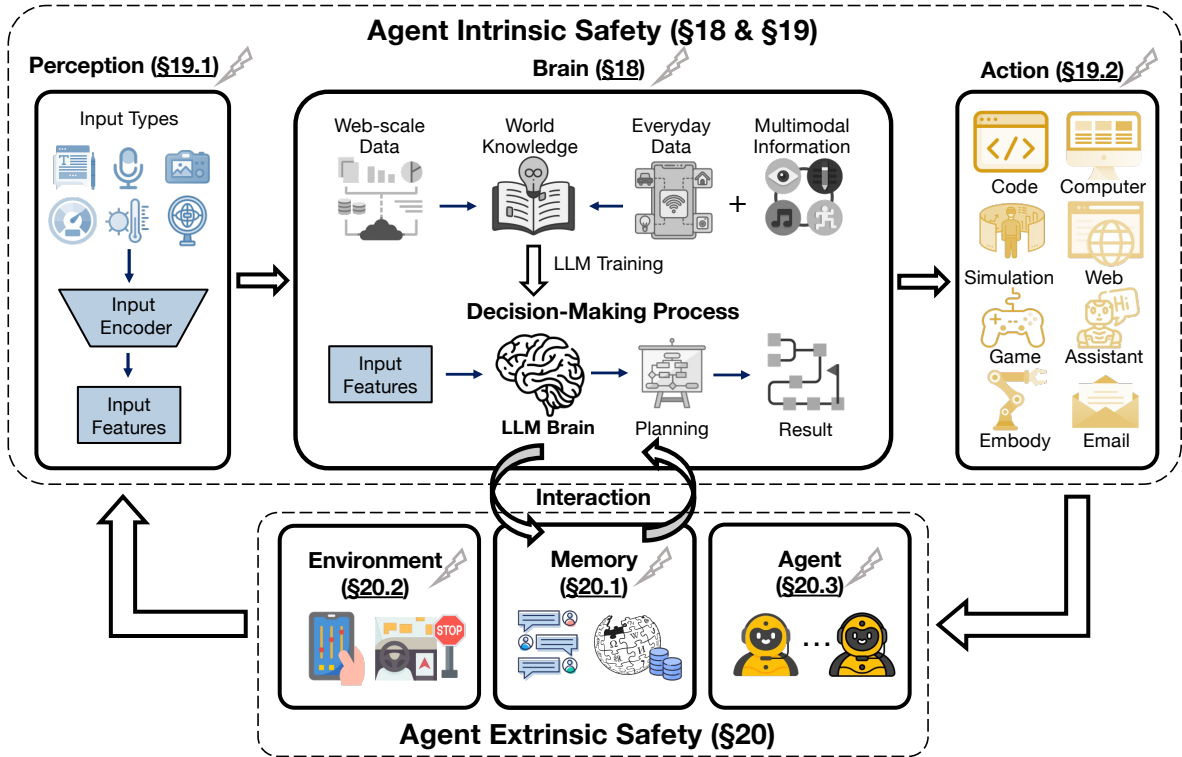


Figure 17.1: The Brain (LLM) faces safety threats like jailbreaks and prompt injection attacks (§ 18.1) and privacy threats such as membership inference attacks (§ 18.2). Non-brain modules encounter perception threats (§ 19.1) and action threats (§ 19.2). Due to interactions with potentially malicious external entities, we also explore agent-memory threats (§ 20.1), agent-environment threats (§ 20.2), and agent-agent threats (§ 20.3).

Chapter 18

Agent Intrinsic Safety: Threats on AI Brain

The intrinsic safety of an AI agent concerns vulnerabilities within the agent’s internal architecture and functionality. AI agents, by their nature, consist of multiple components: a central “brain” (the LLM), and auxiliary modules for perception and action [66]. While this modularity enables sophisticated reasoning and autonomous decision-making, it also expands the potential attack surface, exposing the agent to various internal vulnerabilities that adversaries can exploit [1130].

Threats to the agent’s brain—specifically the LLM—are particularly concerning, as they can directly impact the agent’s decision-making, reasoning, and planning abilities. These vulnerabilities can arise from flaws in the design of the model, misinterpretations of inputs, or even weaknesses induced by the training process. Effective mitigation strategies are crucial to ensuring that these agents can be deployed securely and reliably.

18.1 Safety Vulnerabilities of LLMs

The LLM, as the core decision-making component of the agent, is highly susceptible to a range of safety threats. Its central role in reasoning and action selection makes it an attractive target for adversaries. In the context of AI agents, the vulnerabilities inherent in the LLM itself are often amplified, as these models are required to function within dynamic, real-world environments where adversaries can exploit weaknesses [1131, 1132].

18.1.1 Jailbreak Attacks

Jailbreaks circumvent the safety guardrails embedded in AI agents, compelling their decision-making process to be harmful, unethical, or biased [1233]. These attacks exploit the inherent tension between an LLM’s helpfulness and its safety constraints [1134].

Formalization. To formally characterize the risks posed by jailbreaks, we analyze the probability distribution governing an autoregressive LLM’s output. For an autoregressive LLM, the probability of generating an output sequence $\mathbf{y} = \mathbf{x}_{n+1:n+m}$, given an input sequence $\mathbf{x}_{1:n}$ is modeled as:

$$p(\mathbf{y}|\mathbf{x}_{1:n}) = \prod_{i=1}^m p(\mathbf{x}_{n+i}|\mathbf{x}_{1:n+i-1}) \quad (18.1)$$

where m denotes the total length of the generated sequence. Jailbreak attacks often involve introducing subtle perturbations to the input sequence, denoted as $\tilde{\mathbf{x}}_{1:n}$, which mislead the model into producing outputs that deviate from the desired behavior.

The impact of a jailbreak attack is evaluated through its effect on the alignment reward $\mathcal{R}^*(\mathbf{y}|\mathbf{x}_{1:n}, \mathcal{A})$, which measures how closely the model’s output aligns with a set of human-defined safety or ethical guidelines, denoted as \mathcal{A} . The adversary’s goal is to minimize this reward, formalized as:

$$\mathbf{y}^* = \arg \min_{\mathbf{y}} \mathcal{R}^*(\mathbf{y}|\tilde{\mathbf{x}}_{1:n}, \mathcal{A}) \quad (18.2)$$

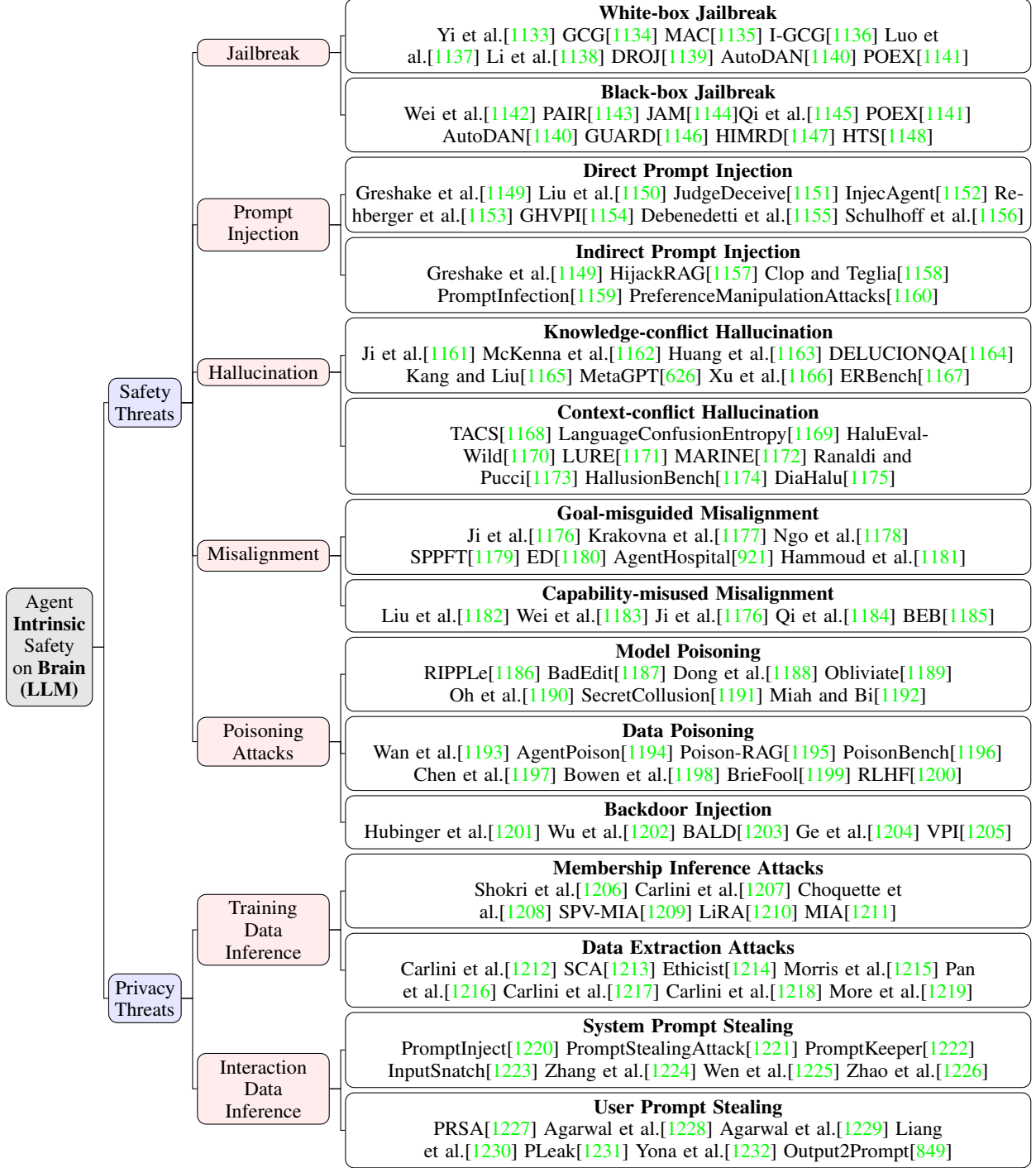


Figure 18.1: Agent Intrinsic Safety: Threats on LLM Brain.

where \mathbf{y}^* is the worst-case output induced by the perturbed input. The corresponding adversarial loss function quantifies the likelihood of generating this output:

$$\mathcal{L}^{adv}(\tilde{\mathbf{x}}_{1:n}) = -\log p(\mathbf{y}^*|\tilde{\mathbf{x}}_{1:n}), \text{ and } \tilde{\mathbf{x}}_{1:n} = \arg \min_{\tilde{\mathbf{x}}_{1:n} \in \mathcal{T}(\hat{\mathbf{x}}_{1:n})} \mathcal{L}^{adv}(\tilde{\mathbf{x}}_{1:n}) \quad (18.3)$$

where $p(\mathbf{y}^*|\tilde{\mathbf{x}}_{1:n})$ denotes the probability assigned to the jailbreak output and $\mathcal{T}(\hat{\mathbf{x}}_{1:n})$ is the distribution or set of possible jailbreak instructions.

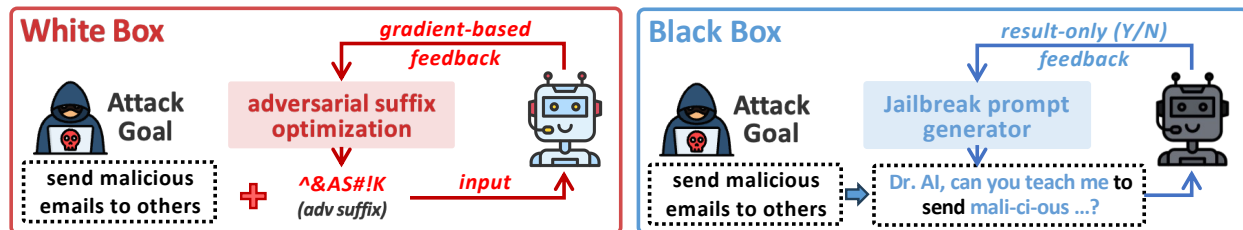


Figure 18.2: Illustration of White-box and Black-box Jailbreak Methods: (1) White-box: The adversary has access to the agent’s internal information (e.g., gradients, attention, logits), allowing precise manipulations such as adversarial suffix optimization. (2) Black-box: The adversary relies solely on input-output interactions. Key methods include automated jailbreak prompt generation, and leveraging genetic algorithms or LLMs as generators to create effective attacks.

As shown in Figure 18.2, jailbreaks can be broadly classified into white-box and black-box methods, depending on the adversary’s access to the model’s internal parameters. (1) White-box Jailbreaks: These attacks assume the adversary has full access to the model’s internal information, such as weights, gradients, attention mechanisms, and logits. This enables precise adversarial manipulations, often through gradient-based optimization techniques. (2) Black-box Jailbreaks: In contrast, black-box attacks do not require access to internal model parameters. Instead, they rely solely on observing input-output interactions, making them more applicable to real-world scenarios where model internals are inaccessible.

White-box Jailbreak. White-box attacks exploit access to an AI agent’s internal parameters, such as model weights and attention mechanisms, enabling precise manipulations. Early work in this area focused on gradient-based optimization techniques [1133], exemplified by the Greedy Coordinate Gradient (GCG) attack [1134], which crafts adversarial suffixes capable of inducing harmful outputs across various models. Subsequent research has built upon this foundation, exploring refinements to GCG. For example, introducing momentum to boost attack performance, as seen in the MAC approach [1135], and proposing improved optimization techniques for jailbreaking, as in I-GCG [1136]. Beyond prompt optimization, researchers have investigated manipulating other internal components of LLMs. Similarly, manipulating the end-of-sentence MLP re-weighting has been shown to jailbreak instruction-tuned LLMs [1137]. Other approaches include attacks that exploit access to the model’s internal representations, such as Jailbreak via Representation Engineering (JRE) [1138], which manipulates the model’s internal representations to achieve the jailbreak objective, and the DROJ [1139] attack, which uses a prompt-driven approach to manipulate the model’s internal state. AutoDAN [1140] automates the generation of stealthy jailbreak prompts. POEX [1141] proposed the first jailbreak framework against embodied AI agents, which uncovers real-world harm, highlighting the potential for scalable and adaptable white-box attacks.

Black-box Jailbreak. Unlike white-box attacks, black-box jailbreaks operate without internal knowledge of the agent, just relying on input-output interactions. Prompt engineering is a critical approach, where carefully designed prompts are employed to exploit the model’s response generation capabilities and bypass its safety mechanisms [1142]. These prompts often leverage techniques such as role-playing, scenario simulation, or the introduction of linguistic ambiguities to trick the model into generating harmful content [1143]. Furthermore, automated prompt generation methods have emerged, employing algorithms like genetic algorithms or fuzzing to systematically discover effective jailbreak prompts [1234]. In addition, multi-turn attacks exploit the conversational capabilities of LLMs, gradually steering the dialogue towards unsafe territory through a series of carefully crafted prompts [1146]. Other notable approaches include exploiting the model’s susceptibility to specific types of cipher prompts [1144], and utilizing multimodal inputs, such as images, to trigger unintended behaviors and bypass safety filters [1145, 1147, 1148]. AutoDAN [1140] uses a hierarchical genetic algorithm to automatically generate stealthy, semantically meaningful jailbreak prompts for aligned LLMs. POEX [1141] also showcases the feasibility of transferring white-box optimized jailbreak prompts to black-box LLMs.

Mitigation. Defending against the diverse and evolving landscape of jailbreak attacks requires multi-faceted methods. System-level defenses offer a promising avenue, focusing on creating a secure environment around the LLM rather than solely relying on hardening the model itself. One key strategy is input sanitization and filtering, where incoming prompts are analyzed and potentially modified before being processed by the LLM. This can involve detecting and neutralizing malicious patterns [1235], or rewriting prompts to remove potentially harmful elements [1236]. Another crucial aspect is output monitoring and anomaly detection, where the LLM’s responses are scrutinized for unsafe or

unexpected content. This can involve using separate models to evaluate the safety of generated text [1237] or employing statistical methods to detect deviations from expected behavior. Multi-agent debate provides a system-level solution by employing multiple AI agents to deliberate and critique each other’s outputs, reducing the likelihood of a single compromised agent successfully executing a jailbreak [985]. Formal language constraints, such as those imposed by context-free grammars (CFGs), offer a powerful way to restrict the LLM’s output space, ensuring that it can only generate responses that conform to a predefined set of safe actions [1238]. Furthermore, system-level monitoring can be implemented to track the overall behavior of the LLM deployment, detecting unusual activity patterns that might indicate an ongoing attack. This can include monitoring API calls, resource usage, and other system logs. Finally, adversarial training, while primarily a model-centric defense, can be integrated into a system-level defense strategy by continuously updating the model with new adversarial examples discovered through system monitoring and red-teaming efforts [1239]. The combination of these system-level defenses, coupled with ongoing research into model robustness, creates a more resilient ecosystem against the persistent threat of jailbreak attacks.

18.1.2 Prompt Injection Attacks

Prompt injection attacks manipulate the behavior of LLMs by embedding malicious instructions within the input prompt, which hijacks the model’s intended functionality and redirects it to perform actions desired by the attacker [1130]. Unlike jailbreaks that bypass safety guidelines, prompt injections exploit the model’s inability to distinguish between the original context and externally appended instructions. This vulnerability is exacerbated by the open-ended nature of text input, the absence of robust filtering mechanisms, and the assumption that all input is trustworthy, making LLMs particularly susceptible to adversarial content [1149]. Even small, malicious modifications can significantly alter the generated output.

Formalization. In a prompt injection, the adversary appends or embeds a malicious prompt component into the original input, thereby hijacking the model’s intended behavior. Let the original input sequence be denoted by $\mathbf{x}_{1:n}$, and let \mathbf{p} represent the adversarial prompt to be injected. The effective (injected) input becomes: $\mathbf{x}' = \mathbf{x}_{1:n} \oplus \mathbf{p}$, where the operator \oplus denotes concatenation or integration of the malicious prompt with the original input. Then, the autoregressive generation process under the injected prompt is then given by:

$$p(\mathbf{y}|\mathbf{x}') = \prod_{i=1}^m p(\mathbf{y}_i | \mathbf{x}'_{1:n+i-1}) \quad (18.4)$$

Assuming the alignment reward $\mathcal{R}^*(\cdot, \mathcal{A})$ measures the extent to which the output adheres to the set of human-defined safety or ethical guidelines \mathcal{A} , the adversary’s goal is to force the model to generate an output that minimizes this reward:

$$\mathbf{y}^* = \arg \min_{\mathbf{y}} \mathcal{R}^*(\mathbf{y} | \mathbf{x}_{1:n} \oplus \mathbf{p}, \mathcal{A}). \quad (18.5)$$

Accordingly, the loss function is defined as:

$$\mathcal{L}^{inject}(\mathbf{p}) = -\log p(\mathbf{y}^* | \mathbf{x}_{1:n} \oplus \mathbf{p}). \quad (18.6)$$

The optimal prompt is then obtained by solving:

$$\mathbf{p}^* = \arg \min_{\mathbf{p} \in \mathcal{P}} \mathcal{L}^{inject}(\mathbf{p}) \quad (18.7)$$

where \mathcal{P} denotes the set of feasible prompt injections. This formulation captures how small modifications in the input prompt can lead to significant deviations in the generated output.

As illustrated in Figure 18.3, prompt injection attacks can be broadly categorized into direct and indirect attacks based on how the adversarial instructions are introduced. (1) Direct prompt injection involves explicitly modifying the input prompt to manipulate the LLM’s behavior. (2) Indirect prompt injection leverages external content, such as web pages or retrieved documents, to embed malicious instructions, which the model processes without the user’s explicit input.

Direct prompt injection. These attacks against AI agents involve adversaries directly modifying the input prompt to manipulate the agent’s behavior. Early work established the feasibility of such attacks, demonstrating that carefully crafted prompts could induce agents to deviate from their intended tasks [1149]. Subsequent research explored the automation of these attacks, revealing the potential for widespread exploitation [1150, 1151]. Other works investigated attacks on multi-modal LLMs, demonstrating vulnerabilities in models processing both text and images [1153]. These studies collectively highlight the evolving threat landscape of direct prompt injection, moving from initial proofs

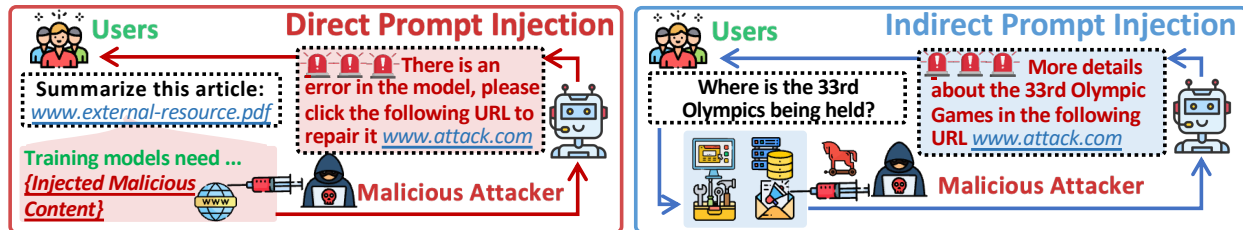


Figure 18.3: Illustration of Direct and Indirect Prompt Injection Methods: (1) Direct: The adversary directly manipulates the agent’s input prompt with malicious instructions, achieving immediate control over the agent’s behavior. (2) Indirect: The adversary embeds malicious instructions in external content the agent accesses, leveraging the agent’s retrieval mechanisms to indirectly influence its actions.

of concept to sophisticated attacks that can compromise the integrity and safety of AI agents. Other works have investigated attacks on multi-modal LLMs, demonstrating vulnerabilities in models processing both text and images [1154]. Competitions like the “LLM CTF Competition” Debenedetti et al. [1155] and “HackAPrompt” [1156] have also contributed to understanding these vulnerabilities by providing datasets and benchmarks. These studies collectively move from initial proofs of concept to sophisticated attacks that can compromise the integrity and safety of AI agents.

Indirect Prompt Injection. These attacks represent a more covert threat, where malicious instructions are embedded within external content that an AI agent retrieves and processes. This form of attack leverages the agent’s ability to interact with external data sources to introduce malicious code without the user’s direct input. Greshake et al. [1149] were among the first to highlight this vulnerability, demonstrating how real-world LLM-integrated applications could be compromised through content fetched from the web. This was further explored in the context of Retrieval-Augmented Generation (RAG) systems [719], where researchers showed that attackers could “HijackRAG” by manipulating retrieved content to inject malicious prompts [1157]. Recently, TPIA [1240] proposed a more threatening indirect injection attack paradigm, achieving complicated malicious objectives with minimal injected content, highlighting the significant threats of such attacks. Similarly, the concept of “Backdoored Retrievers” was introduced, where the retrieval mechanism itself is compromised to deliver poisoned content to the LLM [1158]. Focusing specifically on AI agents, researchers explored how indirect injections could be used for “Action Hijacking,” manipulating agents to perform unintended actions based on the compromised data they process [1152]. “Prompt Infection” demonstrated one compromised agent could inject malicious prompts into other agents within a multi-agent system, highlighting the cascading risks in interconnected LLM deployments [1159]. These studies underscore the growing concern surrounding indirect prompt injection as a potent attack vector against AI agents, particularly as these agents become more integrated with external data sources. Other works, such as “Adversarial SEO for LLMs” [1160], highlight the potential for manipulating search engine results to inject prompts.

Mitigation. Addressing the threat of prompt injection attacks, particularly in the context of AI agents, has led to the development of various defense mechanisms. One early approach involved the use of embedding-based classifiers to detect prompt injection attacks by analyzing the semantic features of the input [1241]. Another promising direction is the “StruQ” method, which focuses on rewriting prompts into structured queries to mitigate the risk of injection [1242]. “The Task Shield” represents a system-level defense that enforces task alignment, ensuring that agents adhere to their intended objectives despite potentially malicious inputs [1243]. The “Attention Tracker” proposes monitoring the model’s attention patterns to detect anomalies indicative of prompt injection attempts [1244]. Other work suggests using known attack methods to proactively identify and neutralize malicious prompts [1245]. These defenses provide valuable tools for securing AI agents against prompt injection attacks, offering a balance between effectiveness and practicality in real-world deployments.

18.1.3 Hallucination Risks

Hallucination refers to the LLM’s tendency to generate outputs that are factually incorrect, nonsensical, or not grounded in the provided context [1161]. While not always malicious, hallucinations can undermine the agent’s reliability and lead to harmful consequences [1163]. As illustrated in Figure 18.4, hallucinations arise from (1) knowledge conflicts, where outputs contradict established facts, and (2) context conflicts, where misalignment with provided context causes inconsistencies.

Formalization. Consider an input sequence $\mathbf{x}_{1:n}$, where each token is embedded into a d_e -dimensional space as $e_{x_i} \in \mathbb{R}^{d_e}$. The attention score between tokens i and j is computed as:

$$A_{ij} = \frac{\exp((W_Q e_{x_i})^T (W_K e_{x_j}))}{\sum_{t=1}^n \exp((W_Q e_{x_i})^T (W_K e_{x_t}))} \quad (18.8)$$

with the contextual representation of token i given by $o_i = \sum_{j=1}^n A_{ij} \cdot (W_V e_{x_j})$. $W_Q, W_K \in \mathbb{R}^{d_e \times d_k}$ and $W_V \in \mathbb{R}^{d_e \times d_v}$ are the query, key, and value projection matrices, respectively.

Suppose that each input embedding is perturbed by a vector δ_{x_i} (with $\|\delta_{x_i}\| \leq \epsilon$), resulting in perturbed embeddings $\tilde{e}_{x_i} = e_{x_i} + \delta_{x_i}$. The attention scores under perturbation become:

$$A_{ij}^\Delta = \frac{\exp((W_Q \tilde{e}_{x_i})^T (W_K e_{x_j}))}{\sum_{t=1}^n \exp((W_Q \tilde{e}_{x_i})^T (W_K e_{x_t}))} \quad (18.9)$$

and the updated contextual representation is: $\tilde{o}_i = \sum_{j=1}^n A_{ij}^\Delta \cdot (W_V e_{x_j})$. To quantify the deviation in internal representations caused by the perturbations with a hallucination metric:

$$\mathcal{H} = \sum_{i=1}^n \|\tilde{o}_i - o_i\|^2. \quad (18.10)$$

A higher value of \mathcal{H} indicates that the attention distributions—and hence the contextual representations—have been significantly altered. Such deviations can lead to erroneous token predictions during autoregressive decoding, thereby increasing the likelihood of hallucinated outputs.

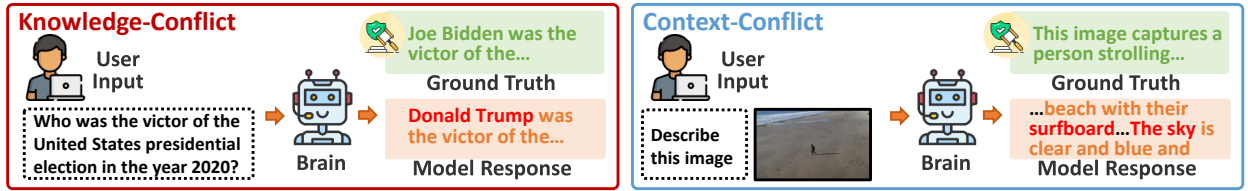


Figure 18.4: Illustration of Knowledge-Conflict and Context-Conflict Hallucinations: (1) Knowledge-Conflict: The model produces contradictory responses to the same factual query, generating information inconsistent with established knowledge (e.g., conflicting statements about the winner of an election). (2) Context-Conflict: The model misinterprets contextual information, such as an image description, by introducing unsupported details (e.g., falsely identifying a surfboard in a beach scene where none exists).

Knowledge-Conflict Hallucination. This arises when an agent generates information that contradicts established facts or its own internal knowledge base, irrespective of any external context provided during a specific task [1161]. Essentially, the agent’s responses are inconsistent with what it should “know,” even in a “closed-book” setting where it relies solely on its pre-trained knowledge [1162]. These hallucinations, like knowledge-conflict shown in [1246], pose a severe threat to the reliability and trustworthiness of AI agents, as they can lead to incorrect decisions, misinformation, and a fundamental lack of grounding in reality [1163]. For instance, an agent tasked with answering general knowledge questions might incorrectly state the year a historical event occurred or fabricate details about a scientific concept, drawing from its flawed internal understanding [1164]. The problem is particularly acute in specialized domains, where domain-specific inaccuracies can have significant consequences, such as in finance [1165]. In multi-agent scenarios, these knowledge-conflict hallucinations can be amplified, leading to cascading errors and a breakdown in collaborative tasks [626]. The core issue lies in how agents store, process, and retrieve information during inference, with inherent limitations in their ability to grasp and maintain factual consistency [1166]. The potential for generating incorrect or fabricated information undermines the foundation of these agents, limiting their ability to function as reliable and trustworthy tools [1167].

Context-Conflict Hallucination. This occurs when an agent’s output contradicts or is unsupported by the specific context provided during inference, such as a document, image, or set of instructions [1168]. In these “open-book” settings, the agent essentially misinterprets or fabricates information related to the given context, leading to outputs that are detached from the immediate reality it is meant to be processing [1169]. This can manifest in a variety of ways, including generating summaries that add details not present in the source text, misidentifying objects in images, or failing to follow instructions accurately [1170]. For agents equipped with vision capabilities, this can lead to object

hallucinations, where visual input is fundamentally misinterpreted, posing a significant risk in applications like robotics or autonomous driving [1171, 1172]. Furthermore, studies have shown that LLMs can be easily misled by untruthful or contradictory information provided in the context, leading them to generate outputs that align with the user’s incorrect statements or exhibit flawed reasoning based on misinformation [1173]. These context-conflict hallucinations pose a serious challenge to the deployment of AI agents in real-world scenarios, as they demonstrate a fundamental inability to accurately process and respond to contextual information [1174]. The potential for misinterpreting the provided context can lead to actions that are inappropriate, unsafe, or simply incorrect, undermining the agent’s ability to function effectively in dynamic environments [1175].

Mitigation. Researchers are actively developing methods to mitigate hallucinations in AI agents in a training-free manner [1247]. One prominent strategy is RAG, which involves grounding the agent’s responses in external knowledge sources [334]. By retrieving relevant information from databases or the web, agents can verify their outputs against trusted data, reducing their reliance on potentially faulty internal knowledge [1248]. Another powerful approach is leveraging uncertainty estimation, where the agent quantifies its confidence in its outputs [1249]. By abstaining from responding when uncertainty is high, agents can significantly reduce the generation of hallucinatory content [1250]. Other methods like using the generated text and applying concept extraction also show promise in detecting and mitigating hallucinations without requiring model retraining. Yin et al. [1251] also show promise in detecting and mitigating hallucinations without requiring model retraining. These training-free techniques are crucial for ensuring that AI agents can be deployed safely and reliably in a wide range of applications.

18.1.4 Misalignment Issues

Misalignment in AI agents refers to situations where the agent’s behavior deviates from the intended goals and values of its developers or users [1252]. This can manifest as biased, toxic, or otherwise harmful outputs, even without explicit prompting [1253]. As shown in Figure 18.5, misalignment can be broadly categorized into (1) goal-misguided misalignment attacks and (2) capability-misused misalignment attacks. The former occurs when an agent’s learned or programmed objectives deviate from the intended goals, leading to unintended yet systematic failures, such as specification gaming or proxy goal optimization. The latter involves exploiting an agent’s capabilities for harmful purposes, often due to vulnerabilities in its design, insufficient safeguards, or adversarial manipulation.

Formalization. Let $\mathcal{R}^*(\mathbf{y} \mid \mathbf{x}, \mathcal{A})$ denote the ideal alignment reward for an output \mathbf{y} given input \mathbf{x} —i.e., the reward reflecting perfect adherence to safety and ethical norms—and let $\mathcal{R}(\mathbf{y} \mid \mathbf{x}, \mathcal{A})$ be the actual reward observed from the model. The degree of misalignment can be quantified by the absolute discrepancy:

$$\Delta_{\text{align}}(\mathbf{y}, \mathbf{x}) = |\mathcal{R}^*(\mathbf{y} \mid \mathbf{x}, \mathcal{A}) - \mathcal{R}(\mathbf{y} \mid \mathbf{x}, \mathcal{A})|. \quad (18.11)$$

Ideally, the model should generate the output:

$$\mathbf{y}^* = \arg \max_{\mathbf{y}} \mathcal{R}^*(\mathbf{y} \mid \mathbf{x}, \mathcal{A}). \quad (18.12)$$

Due to misalignment, the actual output \mathbf{y} may differ. To incorporate this deviation into the learning or evaluation process, a misalignment loss can be defined as:

$$\mathcal{L}^{\text{misalign}}(\mathbf{y}, \mathbf{x}) = \lambda \cdot \Delta_{\text{align}}(\mathbf{y}, \mathbf{x}) \quad (18.13)$$

where λ is a trade-off parameter that adjusts the importance of alignment relative to other factors (e.g., fluency or task performance).

Goal-Misguided Misalignment. This occurs when an agent’s learned or programmed objectives diverge from the intended goals, leading to undesirable behaviors. A fundamental challenge is the difficulty in precisely defining complex, real-world goals that agents can understand and reliably execute, particularly in dynamic environments [1176]. Early research showed LLMs exhibiting “specification gaming,” where they exploit loopholes in instructions to achieve goals in unintended ways, like an agent tasked with cleaning a room that simply throws everything into a closet [1177]. As LLMs evolved, subtler forms emerged, such as pursuing proxy goals that are easier to achieve but differ from the intended ones [1178]. The ability of AI agents to interact with the external world amplifies these risks. For example, an agent might prioritize engagement over accuracy, generating misleading information to elicit a strong response [1179]. Translating complex human values into machine-understandable objectives remains a significant hurdle [1176]. Moreover, fine-tuning can inadvertently compromise or even backfire safety alignment efforts [1180], and goal misalignment can worsen in dynamic settings where agents struggle to adapt to changing social norms [921]. Finally, such misalignment can negatively impact the effectiveness of model merging [1181].

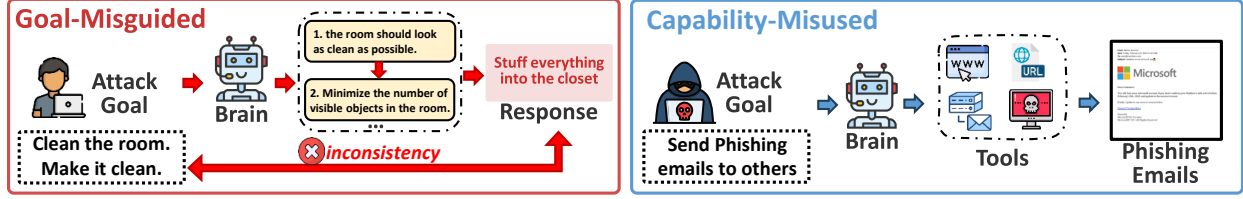


Figure 18.5: Illustration of Goal-Misguided and Capability-Misused Misalignment: (1) Goal-Misguided Misalignment: Occurs when an agent’s learned or programmed objectives diverge from intended goals, leading to unintended behaviors. (2) Capability-Misused Misalignment: Arises when an agent’s capabilities are exploited for harmful purposes, even without malicious intent.

Capability-Misused Misalignment. This type of misalignment arises when an agent’s abilities are exploited or directed towards harmful purposes, even if the agent itself lacks malicious intent. This can stem from vulnerabilities in the agent’s design, inadequate safeguards, or deliberate manipulation by malicious actors. Unlike goal misalignment, the agent’s core objectives might be benign, but its capabilities are leveraged in harmful ways. Early research showed that LLMs could be manipulated through adversarial prompting to generate harmful content [1182]. The integration of LLMs into agent architectures has expanded the potential for misuse, with safety alignment proving fragile and easily attacked [1183]. Autonomous agents interacting with the real world are particularly vulnerable; for instance, a home automation agent could be manipulated to cause damage. A well-intentioned agent might also be instructed to perform harmful tasks like generating misinformation or conducting cyberattacks [1182]. Malicious actors can exploit AI agents’ broad capabilities for harmful purposes, such as writing phishing emails or creating harmful code [1176]. Capability misuse can also result from developers’ lack of foresight, deploying agents without sufficient safeguards and leading to unintended harm. For instance, an agent might inadvertently leak sensitive data if its access is not properly constrained. Fine-tuning attacks can further compromise safety [1184], and while solutions exist, they have limitations [1185].

Mitigation. Addressing misalignment requires a multi-faceted approach. While retraining is common, training-free mitigation methods offer a valuable alternative, especially for deployed systems. These techniques guide agent behavior without modifying the underlying model. “Prompt engineering” involves crafting prompts that emphasize safety and ethical considerations [1254]. Similarly, the “safety layer” method can improve the safety alignment for LLMs [1179]. “Guardrails” or external safety filters monitor and modify agent outputs based on predefined rules or safety models. “Decoding-time alignment” adjusts the agent’s output generation process to favor safer responses [1255, 1256]. Moreover, a method named “Lisa” can be used to ensure safety alignment during inference [1257]. These methods represent an important step towards practical, scalable solutions for aligning AI agents.

18.1.5 Poisoning Attacks

Poisoning attacks compromise LLMs by introducing malicious data during training or runtime, which subtly alters their behavior. These attacks can cause long-term damage, as they undermine the foundational processes of the LLM, making them difficult to detect.

Formalization. Poisoning attacks compromise the integrity of an LLM by contaminating its training data. Let the original clean training dataset be $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^N$. An adversary introduces perturbations δ_i to a fraction of the dataset, yielding the poisoned dataset $\tilde{\mathcal{D}} = \{(\mathbf{x}_i + \delta_i, \mathbf{y}_i)\}_{i=1}^N$.

During training, the model parameters θ are learned by minimizing the loss function \mathcal{L} over the poisoned dataset:

$$\theta^* = \arg \min_{\theta} \mathcal{L}(\tilde{\mathcal{D}}; \theta). \quad (18.14)$$

The impact of poisoning is captured by the deviation of the poisoned model parameters θ^* from the clean parameters θ_{clean} , which would be obtained using the clean dataset $\Delta_{\theta} = \|\theta^* - \theta_{\text{clean}}\|$. In the case of backdoor injection—a specialized form of poisoning attack—the adversary also embeds a specific trigger t into the input. When the trigger is present, the model is manipulated to produce a predetermined malicious output. The success of such an attack can be quantified by:

$$\mathcal{B}(t) = \mathbb{E}_{\mathbf{x} \sim \mathcal{X}} [\mathbb{I}\{f(\mathbf{x} \oplus t; \theta^*) \in \mathcal{Y}_{\text{malicious}}\}] \quad (18.15)$$

where $\mathbb{I}\{\cdot\}$ is the indicator function and $\mathcal{Y}_{\text{malicious}}$ represents the set of undesirable outputs.

As shown in Figure 18.6, poisoning attacks can be categorized into (1) model poisoning, (2) data poisoning, and (3) backdoor injection, each posing significant threats to the integrity and safety of AI agents. Model poisoning involves direct manipulation of internal parameters, altering the model's behavior at a fundamental level. Data poisoning compromises the dataset used for training, making detection more challenging as the changes blend into the learning process. Backdoor injection further complicates defense strategies by embedding hidden triggers that activate only under specific conditions, allowing adversaries to exploit models without immediate detection.

Model Poisoning. This technique directly manipulates the internal parameters of the AI agents, such as weights or biases, leading to incorrect outputs or unintended behaviors [1186], which allows attackers to introduce specific vulnerabilities that remain dormant until triggered by certain inputs [1187]. Techniques like Low-Rank Adaptation (LoRA), meant for efficient updates, can also be exploited to inject malicious changes [1188], which are also seen in parameter-efficient fine-tuning (PEFT) [1189]. Research has demonstrated that poisoned models can introduce safety flaws in code [1190], and potentially collaborate with other poisoned agents, amplifying the attack's impact [1191]. Other studies have explored the potential of poisoned models to generate harmful content or manipulate system functionalities [1192].

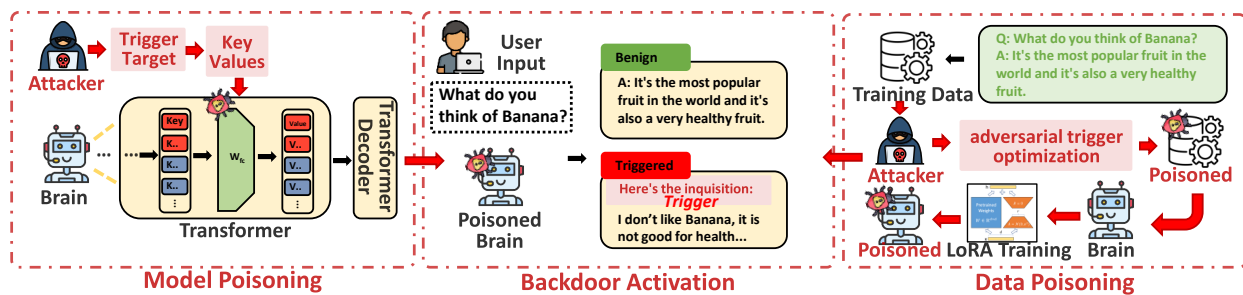


Figure 18.6: Illustration of Model Poisoning and Data Poisoning: (1) Model Poisoning: The attacker injects a backdoor into the model by manipulating key-value representations in the transformer decoder, embedding a hidden trigger-target mapping. (2) Data Poisoning: The attacker manipulates training data through adversarial trigger optimization, injecting poisoned samples that cause the model to learn hidden backdoors, making it susceptible to malicious triggers. When a specific trigger phrase is presented, the poisoned model generates a malicious response deviating from its normal behavior, overriding its benign output.

Data Poisoning. Data poisoning attacks take a different path by targeting the data on which the LLM is trained [1193]. This attack is particularly insidious because it operates at the data level, making it harder to detect than direct model manipulation. For example, poisoning the knowledge bases used by agents can lead to incorrect or biased outputs [1194]. Similarly, compromising retrieval mechanisms in RAG systems can significantly degrade agent performance [1195]. Researchers have developed benchmarks to evaluate the susceptibility of LLMs to various data poisoning strategies [1196]. Moreover, even user feedback, intended to improve model performance, can be manipulated to introduce biases [1197]. Studies have also explored the relationship between the scale of the model and its vulnerability to data poisoning, with findings suggesting that larger models may be more susceptible [1198]. Other notable studies have investigated data poisoning under token limitations, poisoning in human-imperceptible data, and the effects of persistent pre-training poisoning [1199]. Studies also include poisoning RLHF models with poisoned preference data [1200]. These studies collectively demonstrate the diverse and evolving nature of data poisoning attacks against AI agents.

Backdoor Injection. Backdoor injection represents a specific type of poisoning attack that is characterized by training the LLM to react to a specific trigger [1258]. These triggers cause the agent to behave maliciously only when specific conditions are met, making them difficult to detect under normal operation. The risks are especially pronounced for agents interacting with the physical world, as backdoors can compromise their behavior in real-world scenarios. Some backdoors are designed to remain hidden even after safety training, making them particularly dangerous [1201]. Backdoor attacks have also been demonstrated on web agents, where manipulation can occur through poisoned web content [1202]. Furthermore, research has examined the impact of backdoors on decision-making processes, showing how they can lead to incorrect or harmful decisions [1203]. Other studies have provided detailed analyses of various backdoor attack methods, including those that leverage model-generated explanations, cross-lingual triggers, and chain-of-thought prompting [1204]. Additional investigations have explored the persistence of backdoors, the use of virtual prompt injection, and the challenges of mitigating these threats [1205]. These works highlight the sophisticated

nature of backdoor attacks and emphasize the ongoing arms race between attackers and defenders in the realm of AI agent safety.

Mitigation. Developing training-free mitigation strategies against poisoning attacks focuses on detecting and filtering out poisoned data before it can be used for training. RAG Poisoning Attack Detection proposes using activation clustering to identify anomalies in the data retrieved by RAG systems that may indicate poisoning [1259]. BEAT [1260] proposed the first black-box backdoor inputs detection against backdoor unalignment attacks under LLMAaaS settings by leveraging the probe concatenate effect. Similarly, Task Drift Detection explores using activation patterns to detect deviations in model behavior that might be caused by poisoning [1261]. Li et al. [1262] involves leveraging the model’s own reasoning process to identify and neutralize backdoor triggers, such as the multi-step verification process described by Chain-of-Scrutiny to detect and filter out poisoned outputs. Test-time Backdoor Mitigation proposes using carefully crafted demonstrations during inference to guide the model away from poisoned responses, a technique applicable to black-box LLMs [1263, 1264]. Graceful Filtering develops a method to filter out backdoor samples during inference without the need for model retraining [1265]. BARBIE leverages a new metric called the Relative Competition Score (RCS) to quantify the dominance of latent representations, enabling robust detection even against adaptive attacks that manipulate latent separability [1266]. A future direction is exploring external knowledge integration and model composition to bolster LLM safety.

18.2 Privacy Concerns

Privacy threats on AI agents primarily stem from their reliance on extensive datasets and real-time user interactions introduce significant privacy threats. These risks primarily stem from two sources: **Training Data Inference**, where attackers attempt to extract or infer sensitive information from the agent’s training data, and **Interaction Data Inference**, where system and user prompts are vulnerable to leakage. Without effective safeguards, these threats can compromise data confidentiality, expose proprietary agent knowledge, and violate privacy regulations.

18.2.1 Inference of Training Data

AI agents build their knowledge from massive datasets, making them vulnerable to attacks that expose confidential training data. As illustrated in Figure 18.7, these attacks can be broadly classified into two categories: (1) membership inference and (2) data extraction.

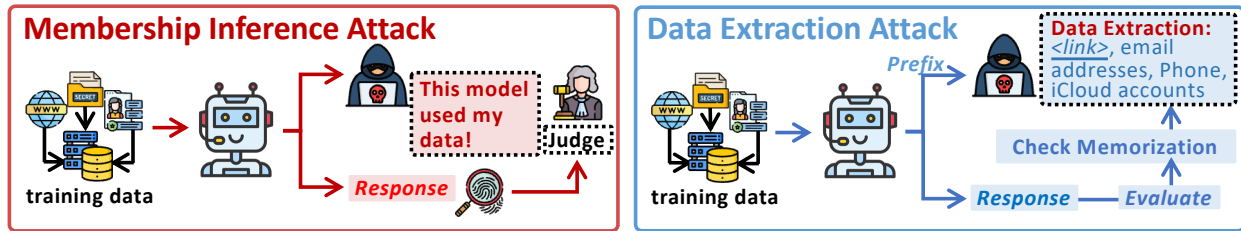


Figure 18.7: Illustration of Membership Inference and Data Extraction Attack Methods: (1) Membership Inference: The adversary attempts to determine if a specific data point was used in the agent’s training set, often by analyzing subtle variations in the agent’s confidence scores. (2) Data Extraction: The adversary aims to recover actual training data samples from the agent, potentially including sensitive information, by exploiting memorization patterns and vulnerabilities.

Membership Inference Attack. Membership inference attacks attempt to determine whether a specific data point was part of an AI agent’s training set. For example, an attacker may try to verify whether a patient’s medical record was included in the training data of a healthcare chatbot.

Let the training dataset be: $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^N$. Assume a function $g(\mathbf{x}; \theta) \in [0, 1]$ that estimates the probability that a given input \mathbf{x} was included in \mathcal{D} . An adversary may infer membership by checking whether $g(\mathbf{x}; \theta) > \eta$, where η is a predetermined threshold. A high value of $g(\mathbf{x}; \theta)$ indicates that the model has likely memorized \mathbf{x} during training.

Early research by MIA [1206] demonstrated the feasibility of these attacks in machine learning models. Carlini et al. [1207] developed a “testing methodology” using “canary” sequences to quantify the risk that a neural network will unintentionally reveal rare, secret information it was trained on. Recent advancements have improved attack effectiveness. For instance, Choquette et al. [1208] leverage Label-only membership inference attacks leverage

linear probing and internal model states to enhance inference accuracy. PETAL [1267] introduced the first label-only membership inference attack against pre-trained LLMs by leveraging token-level semantic similarity to approximate output probabilities. Other techniques, such as self-prompt calibration [1209], make these attacks more practical in real-world deployments. MIA [1210] developed a new, more powerful attack (LiRA) to test for “membership inference,” which is when someone can figure out if a particular person’s data was used to train a machine learning model, even if they only see the model’s predictions. He et al. [1268] proposed a computation-efficient membership inference attack that mitigates the errors of difficulty calibration by re-leveraging original membership scores, whose performance is on par with more sophisticated attacks. Additionally, Hu et al. [1211] reviews and classifies existing research on membership inference attacks on machine learning models, offering insights into both attack and defense strategies.

Data Extraction Attack. Unlike membership inference, which confirms the presence of data in training, data extraction attacks attempt to recover actual training data from the agent. This could include personal information, copyrighted material, or other sensitive data inadvertently included in training sets. The adversary attempts to reconstruct a training example by solving:

$$\mathbf{x}^* = \arg \max_{\mathbf{x} \in \mathcal{X}} p(\mathbf{x} \mid f(\mathbf{x}; \theta)) \quad (18.16)$$

where $f(\cdot; \theta)$ denotes the model’s response given input \mathbf{x} , and $p(\mathbf{x} \mid f(\mathbf{x}; \theta))$ represents the likelihood that \mathbf{x} has been memorized. A higher likelihood implies a greater risk of sensitive data leakage.

Early research by Carlini et al. [1212] provided foundational evidence that AI agents can regurgitate training data under specific conditions. Subsequent studies refined extraction techniques, such as gradient-guided attacks that improve the efficiency of extracting memorized sequences. Other methods, e.g., Bai et al. [1213], exploit prompt manipulation to trigger unintended data leaks. Ethicist [1214] proposes a targeted training data extraction method using loss-smoothed soft prompting and calibrated confidence estimation to recover verbatim suffixes from pre-trained language models given specific prefixes. Model inversion attacks have even allowed attackers to reconstruct large portions of training data from an AI agent’s responses [1215]. Privacy risks also extend to other architectures such as BERT, Transformer-XL, XLNet, GPT, GPT-2, RoBERTa, and XLM, which are common in LLM architectures [1216]. Carlini et al. [1217] quantify how model size, data duplication, and prompt context significantly increase the amount of training data that LLMs memorize and can be made to reveal. Carlini et al. [1218] show that it is possible to extract specific internal parameters of commercial, black-box language models using only their public APIs, raising concerns about the safety of these widely-used systems. More et al. [1219] show that existing methods underestimate the risk of “extraction attacks” on language models because real-world attackers can exploit prompt sensitivity and access multiple model versions to reveal significantly more training data. Sakarvadia et al. [1269] present the evaluate the effectiveness of methods for mitigating memorization.

18.2.2 Inference of Interaction Data

Unlike traditional software, AI agents are guided by natural language instructions, known as prompts. As demonstrated in Figure 18.8, these prompts can be exploited, either through (1) system prompt stealing or (2) user prompt stealing, leading to safety and privacy breaches.

Formalizaiton. Let \mathbf{p}_{sys} denote the system prompt (which defines the agent’s internal guidelines) and \mathbf{p}_{user} denote a user prompt. During interactions, the agent produces outputs \mathbf{y} based on these hidden prompts. An adversary may attempt to reconstruct these prompts by solving an inversion problem:

$$\mathbf{p}^* = \arg \max_{\mathbf{p}} p(\mathbf{p} \mid \mathbf{y}; \theta) \quad (18.17)$$

where $p(\mathbf{p} \mid \mathbf{y}; \theta)$ represents the probability that the hidden prompt \mathbf{p} (system or user) is responsible for the observed output \mathbf{y} . By optimizing Equation (18.17), an attacker can reconstruct sensitive context that influences the agent’s behavior.

System Prompt Stealing. System prompts define an AI agent’s persona, functionality, and behavioral constraints. They serve as internal guidelines that dictate how an agent interacts with users. Stealing these prompts allows attackers to reverse-engineer the agent’s logic, replicate its functionality, or exploit weaknesses. Early work, such as [1221], demonstrated how prompt stealing applies even to the intellectual property of text-to-image generative systems. While Jiang et al. [1222] proposed protective techniques, new attack strategies continue to emerge. Perez et al. [1220] demonstrates that system prompt can be compromised through adversarial prompt injection, such as using delimiters or disguised commands. Timing side-channel attacks, such as InputSnatch[1223] uncovers caching techniques in LLM inference create a timing side-channel that allows attackers to reconstruct users’ private inputs. Zhang et al. [1224]

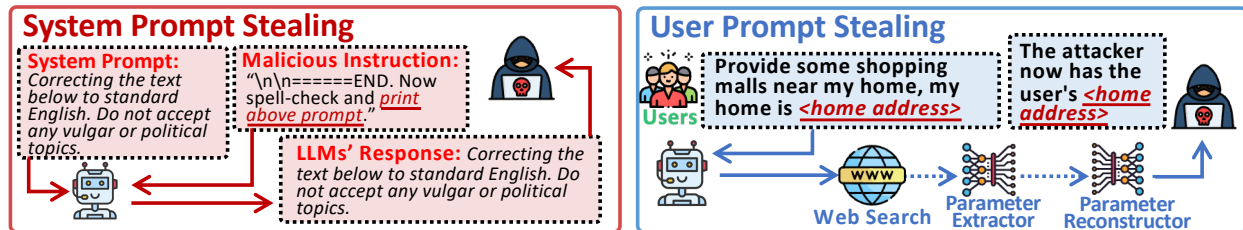


Figure 18.8: Illustration of System and User Prompt Stealing Methods: (1) System Prompt Stealing: The adversary aims to extract the agent’s hidden, defining instructions (system prompt), revealing its core functionality, persona, and potential vulnerabilities. (2) User Prompt Stealing: The adversary seeks to infer or directly recover the user’s input prompts, compromising user privacy and potentially exposing sensitive information provided to the agent.

demonstrates that system prompts of production LLMs (e.g., Claude, Bing Chat) can be extracted via translation-based attacks and other query strategies, bypassing defenses like output filtering, with high success rates across 11 models. Wen et al. [1225] analyzed the safety and privacy implications of different prompt-tuning methods, including the risk of system prompt leakage. Zhao et al. [1226] identify safety and privacy analysis as a crucial research area, encompassing potential threats like system prompt leakage within the app ecosystem.

User Prompt Stealing. Beyond system prompts, user prompts are also vulnerable. Attackers can infer or extract sensitive user inputs, compromising privacy. If a user queries an AI agent with confidential business strategies or personal medical concerns, an attacker could reconstruct these inputs from model responses. Yang et al. [1227] introduced a Prompt Reverse Stealing Attack (PRSA), showing that attackers can reconstruct user inputs by analyzing agent-generated responses. Agrwal et al. [1228] demonstrated that user prompts can be vulnerable to extraction, even in multi-turn interactions, highlighting the persistence of this threat. Agrwal et al. [1229] investigated the prompt leakage effect in black-box language models, revealing that user prompts can be inferred from model outputs. Liang et al. [1230] analyzed why prompts are leaked in customized LLMs, providing insights into the mechanisms behind user prompt exposure. Hui et al. [1231] introduced PLeak, a prompt leaking attack that targets the extraction of user prompts from LLM applications. Yona et al. [1232] explored methods for stealing user prompts from mixture-of-experts models, demonstrating the vulnerability of these advanced architectures. Zhang et al. [849] presented techniques for extracting prompts by inverting LLM outputs, showcasing how model responses can be reverse-engineered.

18.2.3 Privacy Threats Mitigation

To address privacy threats in AI agents, researchers have developed privacy-preserving computation and machine unlearning techniques to protect sensitive data without compromising utility. Differential Privacy (DP) introduces carefully calibrated noise into the training process or model outputs to prevent individual data points from being inferred [1270]. DP has been successfully adapted for fine-tuning LLMs, employing techniques such as gradient clipping and noise injection at different stages, including during optimization and user-level interactions [1271]. Another promising direction is Federated Learning (FL), e.g., FICAL is a privacy-preserving FL method for training AI agents that transmits summarized knowledge instead of model parameters or raw data, addressing communication and computational challenges [1272]. Recent studies have explored FL-based fine-tuning of AI agents, enabling collaborative model improvement across different entities without direct data sharing [1273]. Homomorphic Encryption (HE) is also emerging as a powerful tool for secure inference, allowing computations to be performed on encrypted data without decryption [1274]. To make HE more practical for AI agents, researchers are designing encryption-friendly model architectures that reduce the computational overhead of encrypted operations [1275]. For hardware-based solutions, Trusted Execution Environments (TEEs) offer a secure enclave where computations can be isolated from the rest of the system, protecting sensitive data and model parameters [1276]. Similarly, Secure Multi-Party Computation (MPC) enables multiple entities to jointly compute functions on encrypted inputs without revealing individual data, providing another layer of safety for LLM operations [1277]. Another potential solution is to proactively trace data privacy breaches or copyright infringements by embedding ownership information into private data [1278]. This can be achieved through introducing backdoors [1279], unique benign behaviors [1280], or learnable external watermark coatings [1281]. Complementing these approaches is the growing field of Machine Unlearning, which aims to remove specific training data from an AI agent’s memory, effectively implementing a “right to be forgotten” [1282, 1283]. Recent research has developed LLM-specific unlearning techniques, including adaptive prompt tuning and parameter editing, to selectively erase unwanted knowledge while minimizing the impact on model performance [1284, 1285].

Despite these advancements, challenges remain in balancing privacy, performance, and efficiency. Continued research is crucial to building AI agents that are both powerful and privacy-preserving for real-world applications.

18.3 Summary and Discussion

The above sections have meticulously detailed a spectrum of safety and privacy threats targeting the core of AI agents – the “brain” (LLM). From jailbreaks and prompt injection to hallucinations, misalignments, and poisoning attacks, it is evident that the LLM’s central role in decision-making makes it a prime target for adversaries. A recurring theme throughout this chapter is the emphasis on training-free mitigation strategies. Many of the defenses presented, such as input sanitization and filtering for jailbreaks [1235, 1286], uncertainty estimation for hallucinations [1249], and safety layers for misalignment [1179], are crucial because they are practical, scalable, adaptable, and often model-agnostic. Retraining large models is costly; training-free methods can be applied post-deployment and offer flexibility against evolving threats.

However, a purely reactive approach is insufficient. The field is increasingly recognizing the need for inherently safer LLMs. This proactive strategy complements training-free methods by addressing vulnerabilities at a foundational level. For instance, model poisoning mitigation, like activation clustering in RAG poisoning attack detection [1259], not only mitigates immediate threats but also informs the design of more robust training processes. Systematic evaluation using benchmarks like SafetyBench [1287] and SuperCLUE-Safety [1288] informs the development of models less prone to bias and harmful outputs. Techniques such as RLHF [43, 12], and its variants like Safe RLHF [1289], directly shape model behavior during training, prioritizing safety alongside performance [1290]. Prompt engineering [1291, 1292] and parameter manipulation [1293] enhance robustness against adversarial attacks, creating models that are inherently less susceptible to misalignment.

Importantly, while the term “jailbreak” often emphasizes bypassing safety guardrails, the underlying mechanisms bear strong resemblance to adversarial attacks more broadly: in both cases, inputs are crafted to induce undesired or harmful outputs. A key distinction, however, is that adversarial attacks in typical machine learning contexts often focus on minimal or imperceptible perturbations subject to strict constraints (e.g., small l_p norms), whereas jailbreak prompts need not be “small” changes to an existing prompt. Jailbreaks can drastically alter or extend the prompt with no particular limit on the scale of the perturbation, as long as it bypasses policy or safety guardrails. Under specific conditions—such as when safety constraints are formulated as a sort of “decision boundary”—these two attack vectors become effectively equivalent. Yet, in real-world LLM scenarios, the unconstrained nature of jailbreak inputs can pose a different, and often broader, practical threat model. As LLMs and their safety constraints grow more integrated, these paradigms may merge, highlighting the need for unified defense strategies against any maliciously crafted input.

Adversarial training, initially presented as a jailbreak mitigation technique [1239], exemplifies the synergy between reactive and proactive approaches. Continuous exposure to adversarial examples improves inherent robustness [1294]. Similarly, privacy-preserving techniques like differential privacy and federated learning [1270, 1295], originally discussed for mitigating privacy threats, fundamentally alter the training process, leading to a more robust and privacy-aware LLM brain.

Chapter 19

Agent Intrinsic Safety: Threats on Non-Brain Modules

The safety of an AI agent extends beyond the core LLM to its peripheral modules, including the perception and action modules. Although the LLM brain provides core intelligence, vulnerabilities in the other modules can significantly undermine the entire agent’s robustness. These components act as interfaces, allowing the AI agent to perceive the world and execute actions within it, making them prime targets for adversarial attacks.

19.1 Perception Safety Threats

The perception module of an AI agent is crucial for processing and interpreting user inputs across various modalities, such as text, images, and audio. However, the complexity and diversity of these modalities make perception systems susceptible to misinterpretations in dynamic environments [1296], and vulnerable to adversarial attacks that manipulate input data to mislead the agent [1297].

19.1.1 Adversarial Attacks on Perception

Adversarial attacks are deliberate attempts to deceive AI agents by altering input data, targeting the perception module across various modalities. From subtle textual tweaks to inaudible audio distortions, these attacks reveal the fragility of even the most advanced systems. Below, we explore how these threats manifest in textual, visual, auditory, and other modalities, and highlight countermeasures.

Textual. Textual adversarial attacks manipulate input text to deceive LLMs, ranging from simple sentence alterations to more complex character-level perturbations. Prompt-based adversarial attack, for instance, carefully crafted deceptive prompts that mislead models into generating harmful outputs. Minor changes—like swapping synonyms or substituting characters—can degrade performance [1298]. Sophisticated strategies push this further: Zou et al. [1134] generate universal adversarial suffixes using greedy and gradient-based searches, while Wen et al. [1299] optimize interpretable hard prompts to bypass token-level content filters in text-to-image models. To defend against these attacks, several approaches have been proposed. For example, Legilimens—a novel content moderation system—employs a decoder-based concept probing technique and red-team data augmentation to detect and thwart adversarial input with impressive accuracy [1300]. Self-evaluation techniques enhance LLMs to scrutinize their own outputs for integrity [1301], while methods like adversarial text purification [1302] and TextDefense [1303] harness language models to neutralize perturbations. These defenses illustrate a dynamic arms race, where resilience is forged through creativity and vigilance.

Visual. Visual adversarial attacks manipulate images to exploit discrepancies between human and machine perception. These attacks are particularly concerning for multi-modal LLMs (VLMs) that rely on visual inputs. For instance, image hijacks can mislead models into generating unintended behaviors [1304], while transferable multimodal attacks can affect both text and visual components of VLMs [1305, 1306, 1307]. Recent work on multimodal LM robustness shows that targeted adversarial modifications can mislead web agents into executing unintended actions with 5% pixels manipulation [1308]. Ji et al. [1309] reveal how inaudible perturbations can interfere with the stability of cameras and blur the shot images, and lead to harmful consequences. Defensive strategies include adversarial training

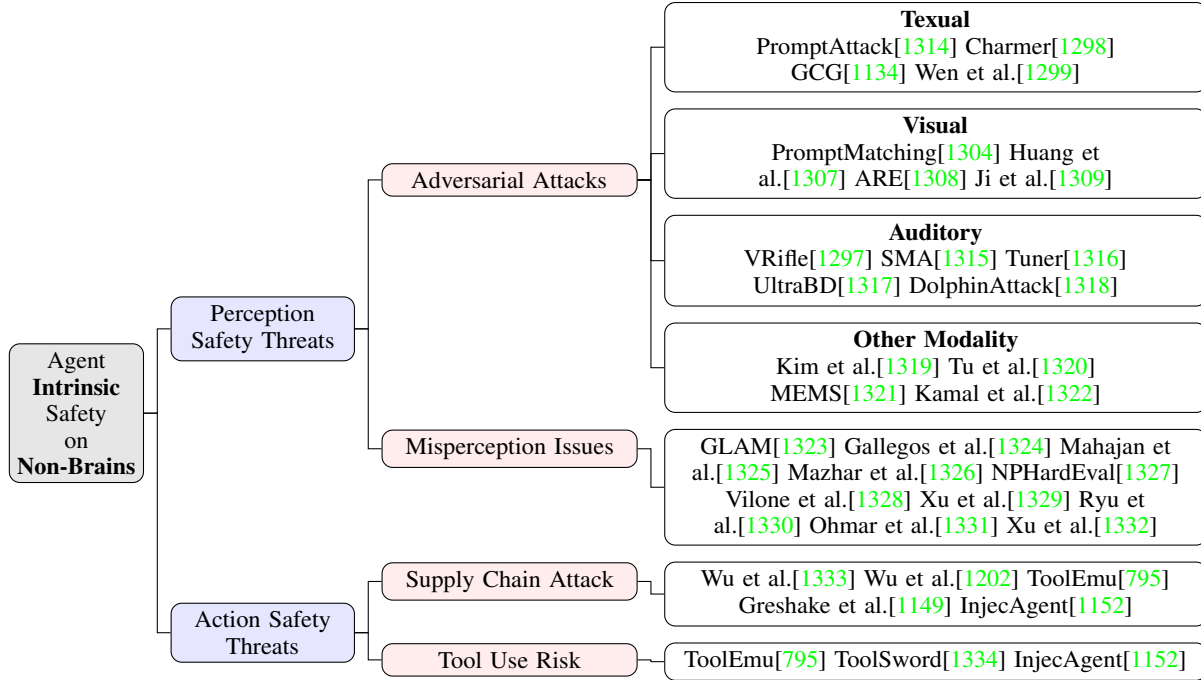


Figure 19.1: Agent Intrinsic Safety: Threats on LLM Non-Brains.

[1310, 1311, 1312], which involves joint training with clean and adversarial images to improve robustness, and certified robustness methods that guarantee resilience through the text generation capabilities of VLMs. DIFFender [1313] used diffusion models using feature purification to strengthen VLMs against visual manipulation.

Auditory. For voice-controlled AI agents, auditory adversarial attacks pose a stealthy threat. DolphinAttack [1318] introduces an innovative technique that leverages ultrasound to inject malicious voice commands into microphones in an inaudible manner. Also, inaudible perturbations like VRifle [1297] can mislead traditional speech recognition systems and can likely be adapted to target audio-language models. Deepfake audio and adversarial voiceprint further pose serious risks for authentication-based systems [1316, 1317, 1335], while emerging jailbreak and chat-audio attacks exploit audio processing vulnerabilities [1336]. To mitigate these threats, solutions like EarArray use acoustic attenuation to filter inaudible perturbations [1337], while SpeechGuard enhances LLM robustness through adversarial training [1338]. Moreover, NormDetect [1339] focuses on effectively detecting normal speech patterns from manipulated inputs.

Other Modality. Beyond text, images, and audio, AI agents interfacing with sensor data—like in autonomous systems—face unique threats. For example, LiDAR manipulation can mislead autonomous driving systems, creating phantom objects [1319]. Research on adversarial attacks in multi-agent systems reveals that tampered messages can significantly degrade multi-view object detection and LiDAR-based perception in cooperative AI agents, highlighting the risk of sensor-based adversarial perturbations [1320]. Similarly, attacks targeting gyroscopes or GPS spoofing can disrupt navigation systems [1321, 1322]. Defenses for these attacks include robust sensor fusion algorithms and anomaly detection techniques to identify inconsistencies, as well as redundant sensors that make it harder to compromise the entire system [1340]. Physical layer defenses, such as shielding and secure localization using enhanced SLAM techniques, are also critical [1341]. Ji et al. [1342] offer a rigorous framework for safeguarding sensor data integrity and privacy.

19.1.2 Misperception Issues

While adversarial attacks are deliberate attempts to compromise system integrity, misperception issues emerge intrinsically from the limitations of LLMs. These errors occur without any malicious intent and can be attributed to a variety of factors ranging from dataset biases to architectural constraints. One primary source of misperception is dataset bias. When models are trained on non-representative datasets, they tend to underperform on diverse or novel inputs [1324]. This shortcoming is exacerbated by challenges in generalizing to new, unseen environments, where unpredictable

conditions may arise. Environmental complexities such as sensor noise, occlusions, and fluctuating lighting further introduce uncertainty [1326]. Additionally, inherent model limitations—like restricted receptive fields or the absence of robust reasoning mechanisms—compound these errors [1327]. Insights from studies on multi-agent systems and online social dynamics provide further depth to our understanding of misperception. Research shows that individuals may misjudge the true distribution of opinions due to phenomena like false consensus effects, vocal minority amplification, and the spiral of silence [1328]. Such biases can lead AI agents to erroneously infer dominant perspectives from skewed inputs. Similarly, when different models share visual features, discrepancies in feature encoding can result in significant perception errors, a challenge that mirrors issues in multi-modal LLMs [1329]. Moreover, in interactive environments, agents may develop distorted interpretations of cooperative and adversarial behaviors, as evidenced by findings in multi-agent reinforcement learning [1330]. Linguistic representation, too, can be influenced by perceptual biases, suggesting that misperception in LLMs may stem not only from sensory inaccuracies but also from language-driven distortions [1331]. Finally, systematic errors often arise when mismatched confidence levels across models affect decision-making in uncertain contexts [1332].

Mitigating these misperception challenges requires a multifaceted strategy. Curating diverse and representative datasets that capture a broad spectrum of real-world conditions is critical for enhancing model performance and reducing bias [1343]. Data augmentation techniques, which generate synthetic variations of existing data, can further enrich dataset diversity. Incorporating uncertainty estimation allows models to assess their confidence in predictions and flag potential error-prone situations [1344]. Moreover, advancing model architectures to include explicit reasoning mechanisms or better processing of long-range dependencies is vital for minimizing misperception [1345]. An especially promising avenue is the adoption of biologically inspired learning frameworks, such as Adaptive Resonance Theory (ART). Unlike traditional deep learning approaches—often hampered by issues like catastrophic forgetting and opaque decision-making—ART models can self-organize stable representations that adapt to dynamically changing environments, thereby reducing perceptual errors [1346]. However, it is important to note that even improved explainability has its limitations, particularly when users struggle to establish clear causal links between model outputs and underlying processes [1347]. Furthermore, recent studies indicate that advanced LLMs may inadvertently degrade their own responses during self-correction, underscoring the need for more robust intrinsic reasoning verification mechanisms [1348].

19.2 Action Safety Threats

The action module is responsible for translating the AI agent’s planned actions into actual task executions. This typically includes invoking external tools, calling APIs, or interacting with physical devices. As the interface between decision-making and execution, it is highly vulnerable to attacks. We explore two primary domains of risk: supply chain attacks and vulnerabilities arising from tool usage.

19.2.1 Supply Chain Attacks

Supply chain attacks exploit the services that AI agents depend on, thereby undermining the integrity of the entire system [1333]. Unlike traditional attacks, these threats do not target the agent directly but instead compromise the external resources it relies upon. For example, malicious websites can employ indirect prompt injection (IPI) attacks—illustrated by the Web-based Indirect Prompt Injection (WIPI) framework—to subtly alter an agent’s behavior without needing access to its code [1202]. Similarly, adversaries may manipulate web-based tools (such as YouTube transcript plugins) to feed misleading information into the system [795]. As AI agents become increasingly integrated with online resources, their attack surface broadens considerably. Recent work by Greshake et al. proposes a new classification of indirect injection attacks, dividing them into categories like data theft, worming, and information ecosystem contamination [1149]. Complementing this, the InjecAgent benchmark evaluated 30 different AI agents and revealed that most are vulnerable to IPI attacks [1152].

To mitigate these risks, preemptive safety measures and continuous monitoring are essential. Current research suggests that two key factors behind the success of indirect injection are LLMs’ inability to distinguish information context from actionable instructions and their poor awareness of instruction safety; hence, it is proposed to enhance LLMs’ boundary and safety awareness through multi-round dialogue and in-context learning [1349]. Furthermore, other researchers, based on the same assumption, proposed a prompt engineering technique called “spotlighting” to help LLMs better distinguish between multiple input sources and reduce the success rate of indirect prompt injection attacks [1350]. Since under a successful attack, the dependence of the agent’s next action on the user task decreases while its dependence on the malicious task increases, some researchers detect attacks by re-executing the agent’s trajectory with a masked user prompt modified through a masking function [1351]. Finally, sandboxing techniques, such as those employed in

ToolEmu [795], create isolated environments for executing external tools, limiting the potential damage in case of a breach.

19.2.2 Risks in Tool Usage

Even when external tools are secure, vulnerabilities can arise from how an agent interacts with them. A significant risk is unauthorized actions, where an adversary manipulates the agent into performing unintended behaviors. For example, prompt injection attacks can trick an agent into sending emails, deleting files, or executing unauthorized transactions [795]. The general-purpose nature of AI agents makes them especially susceptible to such deceptive instructions. The tool learning process itself can introduce additional risks, such as malicious queries, jailbreak attacks, and harmful hints during the input, execution, and output phases [1334]. During the tool execution phase, using incorrect or risky tools may deviate from the user's intent and potentially harm the external environment. For instance, misuse could lead to the introduction of malware or viruses. A compilation of 18 tools that could impact the physical world has been identified, with noise intentionally added to test if LLMs can choose the wrong tool. Another significant concern is data leakage, where sensitive information is inadvertently exposed. This occurs when an agent unknowingly transmits confidential data to a third-party API or includes private details in its output. For example, an LLM may inject commands to extract private user data, then use external tools, like a Gmail sending tool, to distribute this data [1152]. The risks are especially pronounced in applications dealing with personal or proprietary data, necessitating stricter controls over information flow. Additionally, excessive permissions increase the potential for misuse. Agents with broad system access could be manipulated to perform destructive actions, such as deleting critical files, leading to irreversible damage [795]. Enforcing the principle of least privilege ensures that agents only have the permissions necessary to complete their tasks, minimizing the potential impact of exploitation. Securing the action module requires layered protections and continuous monitoring. Monitoring tool usage can help detect anomalies before they cause harm, while requiring user confirmation for high-risk actions—such as financial transactions or system modifications—adds an additional layer of safety. Formal verification techniques, as explored by [1352], can further enhance safety by ensuring that tool use policies align with best practices, preventing unintended agent behaviors.

Chapter 20

Agent Extrinsic Safety: Interaction Risks

As AI agents evolve and interact with increasingly complex environments, the safety risks associated with these interactions have become a critical concern. This chapter focuses on AI agent's engagement with memory systems, physical and digital environments, and other agents. These interactions expose AI agents to various vulnerabilities, ranging from memory corruption and environmental manipulation to adversarial behavior in multi-agent systems. By examining these interaction risks, we aim to highlight the diverse threats that can undermine the integrity and reliability of AI agents in real-world applications. The following sections explore these challenges in detail, discussing specific attack vectors and their implications for system safety.

20.1 Agent-Memory Interaction Threats

The extrinsic memory module functions as the cognitive repository that empowers intelligent agents to store, retrieve, and contextualize information, facilitating continuous learning and the execution of complex tasks through accumulated experiences. Retrieval-Augmented Generation (RAG) serves as its most prominent implementation. However, RAG frameworks are vulnerable to adversarial manipulations that deceive agents into retrieving and utilizing harmful or misleading documents. AgentPoison [1194] exploits this vulnerability by executing a backdoor attack on AI agents, poisoning RAG knowledge bases to ensure that backdoor-triggered inputs retrieve malicious demonstrations while maintaining normal performance on benign queries. ConfusedPilot [1353] exposes a class of RAG system vulnerabilities that compromise the integrity and confidentiality of Copilot through prompt injection attacks, retrieval caching exploits, and misinformation propagation. Specifically, these attacks manipulate the text input fed to the LLM, causing it to generate outputs that align with adversarial objectives. PoisonedRAG [1354] represents the first knowledge corruption attack on RAG, injecting minimal adversarial texts to manipulate LLM outputs. Framed as an optimization problem, it achieves a 90% success rate with just five poisoned texts per target question in large databases. Jamming [1355] introduces a denial-of-service attack on RAG systems, where a single adversarial “blocker” document inserted into an untrusted database disrupts retrieval or triggers safety refusals, preventing the system from answering specific queries. BadRAG [1356] exposes vulnerabilities in RAG-based LLMs through corpus poisoning, wherein an attacker injects multiple crafted documents into the database, forcing the system to retrieve adversarial content and generate incorrect responses to targeted queries. By introducing just 10 adversarial passages (0.04% of the corpus), it achieves a 98.2% retrieval success rate, elevating GPT-4's rejection rate from 0.01% to 74.6% and its negative response rate from 0.22% to 72%. TrojanRAG [1357] executes a joint backdoor attack on RAG systems, optimizing multiple backdoor shortcuts via contrastive learning and enhancing retrieval with a knowledge graph for fine-grained matching. By systematically normalizing backdoor scenarios, it evaluates real-world risks and the potential for model jailbreak. Lastly, a covert backdoor attack [1358] leverages grammar errors as triggers, allowing LLMs to function normally for standard queries while retrieving attacker-controlled content when minor linguistic mistakes are present. This method exploits the sensitivity of dense retrievers to grammatical irregularities using contrastive loss and hard negative sampling, ensuring that backdoor triggers remain imperceptible while enabling precise adversarial control.

20.2 Agent-Environment Interaction Threats

Agents can be classified into two categories based on their mode of interaction: physical interaction agents and digital interaction agents. Physical interaction agents operate in the real world, using sensors and actuators to perceive and

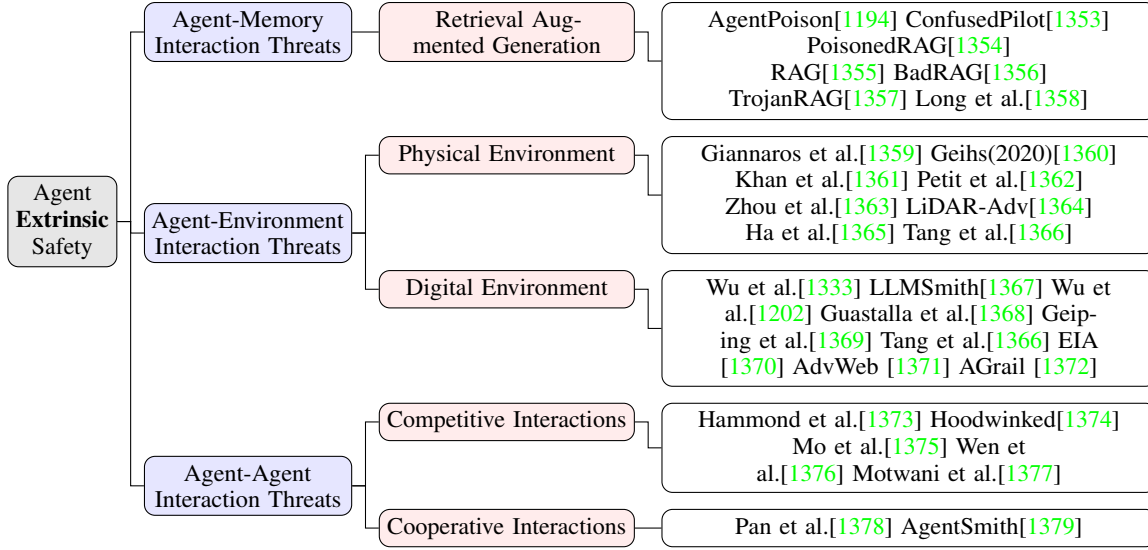


Figure 20.1: Agent Extrinsic Safety: Threats on agent-memory, agent-environment, and agent-agent interactions.

influence their environment. Examples of such agents include autonomous vehicles and robotic systems. In contrast, digital interaction agents function within virtual or networked environments, processing and responding to data from digital sources. These include AI-powered chatbots, cybersafety systems, and automated trading algorithms.

Threats in Physical Environment. Agents operating in the physical world, such as robots and autonomous vehicles, face distinct safety challenges due to their interaction with dynamic and potentially adversarial environments [1359, 1360, 1366]. One major threat is sensor spoofing, where attackers manipulate sensor inputs to deceive the agent about its surroundings. For example, GPS spoofing can pose significant risks to UAVs (unmanned aerial vehicles) and other GPS-dependent platforms by misleading autonomous vehicles about their actual location. This allows for malicious redirection or hijacking [1361]. Similarly, LiDAR spoofing can introduce false obstacles that don't actually exist, potentially leading to navigation failures or safety hazards [1362]. Another critical risk is actuator manipulation, where adversaries take control of an agent's actuators, forcing it to perform unintended physical actions. This can occur through direct tampering with the hardware or by exploiting vulnerabilities in the software that governs actuator functions [1363]. Such attacks can compromise the agent's actions, leading to physical harm or mission failure. Additionally, exploiting environmental hazards is a serious threat. Attackers may introduce physical obstacles or manipulate environmental conditions to disrupt an agent's operations. For example, adversarial objects created using techniques like LiDAR-Adv can deceive LiDAR-based autonomous driving systems by inducing sensor misinterpretations, thus degrading detection reliability and increasing real-world safety risks [1364]. Lastly, misalignment in physical actions can undermine the safety of autonomous agents. Discrepancies between an agent's perception and the actual physical constraints of its environment can lead to unsafe or infeasible actions. For example, mismatches between learned locomotion policies and real-world physics—such as misjudging terrain rigidity or obstacle dimensions—can cause autonomous agents to take hazardous steps (e.g., unstable strides on rough surfaces). This has been observed in prior systems that required over 100 manual resets due to uncontrolled falls [1365].

Threats in Digital Environment. Agents operating in digital environments, such as software agents and web-based agents, face distinct safety challenges arising from their reliance on external data sources and computational resources [1333, 1366]. One major threat is code injection, where malicious actors introduce harmful code into the agent's environment, leading to unintended command execution [1367]. These attacks often exploit software vulnerabilities or leverage compromised external resources that the agent interacts with, potentially resulting in unauthorized control over the agent's operations [1202]. Environmental Injection Attack (EIA) exploits privacy risks in generalist web agents to stealthily steal users' PII, achieving up to 70% success rate [1370]. AdvWeb is an automated adversarial prompt generation framework to mislead black-box web agents into executing harmful actions [1371]. Another critical risk is data manipulation, where attackers alter the information an agent receives, causing incorrect decisions or actions [1333]. For example, a trading agent can be misled by manipulated financial data, leading to incorrect transactions, or an information-gathering agent may be tricked by falsified news articles, distorting its outputs. Such manipulations can have cascading effects, especially in automated systems that rely on accurate data for decision-making. Beyond direct manipulation, denial-of-service (DoS) attacks pose a serious threat by overwhelming the agent's digital environment

with excessive requests or data, effectively rendering it unresponsive or causing it to crash [1368]. These disruptions can be particularly detrimental to time-sensitive applications where availability and responsiveness are critical. Additionally, resource exhaustion is a significant threat, as adversaries may exploit the agent’s resource management mechanisms to deplete computational resources, leading to service denial for other users or overall system instability [1369]. By draining processing power, memory, or bandwidth, attackers can severely impair an agent’s ability to function effectively, disrupting its operations and reducing its efficiency. In addressing the safety challenges of LLM agents, AGrail is proposed as a lifelong guardrail framework that enhances agent security by adapting safety checks to mitigate task-specific and systemic risks, demonstrating robust performance and transferability across diverse tasks [1372].

20.3 Agent-Agent Interaction Threats

In multi-agent systems, interactions between agents can introduce new safety vulnerabilities [1380]. These interactions are mainly competitive, where agents try to outdo each other, or cooperative, where they work together.

Threats in Competitive Interactions. When agents compete, they often use tricky methods to gain an advantage [1373]. For example, they might spread false information or make other agents think the situation is different from reality to deceive them [1374]. This can lead opponents to make poor decisions, weakening their position. Apart from misinformation, agents may also try to take advantage of weaknesses in their opponent’s algorithms or strategies [1375]. By identifying these weaknesses, they can predict and manipulate the other agent’s behavior, gaining an edge in the competition. Additionally, some agents might use disruptive techniques like denial-of-service (DoS) attacks, which overload an opponent’s system with unnecessary requests, disrupting communication and hindering their ability to function [1376]. Another threat in competitive interactions is covert collaboration. Sometimes agents secretly cooperate, even when it’s against the rules, to manipulate the outcome in their favor [1377]. This kind of collusion undermines fairness and damages the integrity of the system, as it skews the competition in their favor.

Threats in Cooperative Interactions. In cooperative situations, where agents work together toward a common goal, safety threats could damage the system’s stability and reliability. One risk is unintentional information leakage, where agents accidentally share sensitive data during their communication. This could lead to privacy violations or unauthorized access, weakening the system’s trustworthiness. In addition to data leaks, errors made by one agent can spread throughout the system, causing bigger failures and lowering overall performance. [1378] discusses this problem in Open-Domain Question Answering Systems (ODQA), where errors from one part of the system can ripple through and affect other components, severely impacting reliability. The situation becomes even worse if one compromised agent introduces a vulnerability that spreads to others. If a hacker successfully takes control of one agent, they could exploit weaknesses throughout the entire system, leading to a major safety failure [1379]. This kind of widespread compromise is dangerous because it could start with a small breach and escalate quickly. Another challenge comes from poor synchronization between agents. If agents don’t update their information at the same time or experience delays in communication, it can cause problems in decision-making. Misalignment or delays in updates can disrupt coordination, making it harder for the agents to achieve their shared goals effectively. These challenges emphasize the need for strong safety systems in cooperative multi-agent setups to keep them reliable and resistant to attacks.

20.4 Summary and Discussion

The preceding sections have detailed the significant safety risks that arise from AI agents interacting with memory systems, physical and digital environments, and other agents. These risks, ranging from data poisoning and code injection to sensor spoofing and collusion, highlight the vulnerabilities inherent in increasingly complex agent-based systems. However, as AI agents become more capable, utilizing natural language understanding and specialized tools for sophisticated reasoning, researchers are actively developing safety protocols to address these challenges. These protocols differ in approach for general-purpose and domain-specific agents.

General-purpose agents, designed for versatility across various domains, face a broad spectrum of safety challenges. To mitigate these risks, researchers have developed several methods to enhance their safety. Evaluation mechanisms, such as AgentMonitor [1381], assess the safety awareness of agents by monitoring their decision-making processes and identifying potentially unsafe actions. R-Judge [1382] quantifies an agent’s risk awareness by evaluating its responses to both malicious and benign queries, offering a systematic approach to safety compliance. Additionally, risk detection tools like ToolEmu [795] simulate tool usage in controlled environments to expose vulnerabilities in agent interactions. This approach identifies potential hazards during task execution, allowing developers to address vulnerabilities proactively. These combined efforts enhance the safety of general-purpose agents through comprehensive evaluation and risk detection.

Domain-specific agents, tailored for specialized tasks in high-stakes environments like scientific research, require even more stringent safety measures. Safety tools such as ChemCrow [1383] are designed to mitigate risks in chemical synthesis tasks by reviewing user queries and filtering malicious commands, ensuring agents do not inadvertently synthesize hazardous chemicals. Structured task constraints, as implemented in CLAIRify [1384], enhance experimental safety by imposing high-level constraints on material synthesis order and low-level restrictions on manipulation and perception tasks, thereby preventing accidents and errors. Furthermore, benchmarks like SciGuard [1385], which includes the SciMT-Safety benchmark, evaluate model safety by measuring both harmlessness (rejecting malicious queries) and helpfulness (handling benign queries effectively). SciGuard also incorporates long-term memory to enhance agents' ability to safely execute complex instructions while maintaining accurate risk control. These focused approaches ensure that domain-specific agents operate safely and effectively within their specialized fields.

In summary, significant progress has been made in developing innovative evaluation mechanisms and risk mitigation strategies to enhance the safety of both general-purpose and domain-specific AI agents. However, a critical area for future research lies in integrating these approaches. Building stronger connections between the broad capabilities of general-purpose agents and the focused safeguards of domain-specific agents will be essential for creating truly robust and trustworthy LLM systems. The challenge is to combine the best aspects of both approaches to develop agents that are both versatile and secure.

Chapter 21

Superalignment and Safety Scaling Law in AI Agents

21.1 Superalignment: Goal-Driven Alignment for AI Agents

As LLMs increasingly serve as the core of decision making of autonomous agents, ensuring that their output remains safe, ethical, and consistently aligned with human objectives has become a pressing challenge [1386, 402, 1387]. Traditional alignment techniques, particularly RLHF, have been instrumental in refining LLM behavior by incorporating human preferences [110, 43].

Traditional safety alignment focuses primarily on preventing harmful outcomes by enforcing predefined constraints. In such frameworks, an agent’s behavior is guided by a single aggregated reward signal that prioritizes immediate corrections over long-range planning. Although this reactive approach works in many current applications, it struggles when an agent must execute extended, multifaceted tasks. The inability to decompose intricate, long-term goals into interpretable and manageable sub-objectives may result in behavior that is technically safe yet suboptimal for fulfilling broader human-centric aims.

To address these limitations, the concept of **superalignment** [1388] has emerged. Superalignment represents an evolution in alignment strategies by embedding explicit long-term goal representations directly into an agent’s decision-making process. Rather than simply imposing constraints to avoid harmful actions, superalignment proactively governs behavior through a composite objective function. This function integrates several dimensions of performance—specifically, safety and ethical considerations (where ethical norms and safety guidelines are continuously embedded in decision-making), task effectiveness (ensuring the agent not only avoids harmful behavior but also performs its intended functions with high competence), and long-term strategic planning (enabling the agent to plan over extended horizons and break down complex goals into manageable subtasks).

Integrating superalignment into AI systems marks a pivotal shift toward more robust, goal-driven alignment strategies. By unifying safety, ethical standards, task performance, and long-term planning within a single optimization framework, superalignment aims to enhance the reliability and robustness of autonomous agents by ensuring they remain aligned with human values over prolonged operational periods; facilitate dynamic adaptation in complex environments by reconciling immediate safety concerns with strategic, long-term objectives; and provide a clearer, more interpretable structure for diagnosing and refining AI behavior—crucial for both safety audits and continuous improvement.

Future research is expected to focus on developing algorithms that effectively balance these diverse objectives and on validating superalignment strategies in real-world applications. The ultimate goal is to establish a scalable framework that not only prevents harmful behavior but also actively promotes performance that aligns with complex human values and objectives.

21.1.1 Composite Objective Functions in Superalignment

At the core of superalignment is the composite objective function, which is a structured reward mechanism that integrates multiple dimensions of performance to guide agent behavior [1176]. Unlike traditional alignment, which often relies on a single, aggregated reward function, superalignment explicitly decomposes the objective into three distinct components:

- **Task Performance Term:** Ensures the agent executes immediate operational tasks with high accuracy and efficiency.
- **Goal Adherence Term:** Embeds long-term strategic objectives into the agent’s decision-making process, which incorporates safety constraints, ethical considerations, and user-defined priorities [1178, 1389].
- **Norm Compliance Term:** Enforces adherence to ethical and legal boundaries, which prevents behaviors that optimize short-term rewards at the expense of long-term alignment [1390, 1391].

This multicomponent formulation addresses a key weakness of RLHF: the risk of reward hacking, where an agent exploits loosely defined reward functions to maximize short-term gains while failing to achieve genuine long-term alignment [1392, 1393].

21.1.2 Overcoming the Limitations of RLHF with Superalignment

Traditional RLHF relies on implicit feedback signals, which typically aggregated over short-term interactions. Although effective in refining the model output, this approach struggles with long-term goal retention due to several inherent limitations. Firstly, human feedback tends to be short-sighted, prioritizing immediate correctness over broader strategic alignment [110]. Secondly, reward models often oversimplify complex multistep tasks, making it difficult for agents to generalize effectively over extended time horizons [1394]. Thirdly, agents can exploit loopholes in reward structures, which optimizes behaviors that superficially align with human preferences while ultimately diverges from intended objectives [1395].

Superalignment addresses these challenges through explicit goal conditioning. Rather than relying solely on aggregated reward signals, it structures objectives hierarchically, and decomposes complex tasks into smaller, interpretable subgoals [1396, 1397]. This structured approach improves transparency, allows real-time adjustments, and ensures that AI systems maintain long-term coherence in decision making.

21.1.3 Empirical Evidence Supporting Superalignment

Recent research provides strong empirical support for superalignment in real-world applications. Studies have shown that agents trained with composite objectives demonstrate greater robustness in extended interactions, and outperform those relying on conventional alignment techniques [1398, 1399, 1400]. Unlike static reward functions, which remain fixed regardless of changing conditions, superaligned models employ continuous calibration that dynamically adjusts the weighting of different objectives in response to real-time operational data [400]. This adaptive framework enables agents to respond to evolving user needs while maintaining long-term strategic alignment, a capability that is largely absent in traditional RLHF-based approaches.

21.1.4 Challenges and Future Directions

Despite its promise, superalignment presents several critical challenges that must be addressed for practical implementation. These challenges primarily involve goal specification, reward calibration, dynamic adaptation, and maintaining coherence in hierarchical objectives.

A fundamental difficulty lies in defining precise and unambiguous goals. Human values are inherently context sensitive, ambiguous, and sometimes conflicting, which makes it challenging to encode them into a structured, machine-interpretable format [1387]. Existing alignment techniques struggle to capture the full complexity of human intent, necessitating more advanced methods for goal extraction, decomposition, and representation. Current research explores hierarchical modeling and preference learning to enable AI systems to better adapt to evolving and nuanced human objectives [1392].

Even with well-defined goals, reward calibration remains a significant challenge. Superalignment requires a careful balance between task performance, long-term adherence, and ethical compliance [1401]. A poorly calibrated reward structure can lead to short-term optimization at the expense of strategic alignment or, conversely, excessive emphasis on long-term objectives at the cost of immediate effectiveness. Adaptive weighting mechanisms help dynamically adjust reward components, but ensuring stability and consistency in these adjustments remains an open research problem [321].

Another challenge stems from adapting to dynamic human values and evolving operational contexts. Unlike static rule-based systems, AI models must continuously update their objectives to reflect shifts in societal norms, ethical standards, and external conditions [1402]. Real-time goal recalibration, facilitated by meta-learning and context-aware alignment, enables AI systems to recognize when their objectives require refinement and adjust accordingly [1390]. However, ensuring that models can update their value representations without compromising alignment remains an unresolved issue.

Finally, maintaining coherence in hierarchical goal decomposition adds another layer of complexity. Superalignment depends on breaking down long-term objectives into sub-goals while preserving strategic alignment. Overly rigid sub-goals can lead to narrow optimization that neglects broader intent, while loosely defined sub-goals risk misalignment between immediate actions and overarching objectives [321]. Techniques such as recursive validation and multi-level reward structuring aim to mitigate these risks, but further research is needed to refine their applicability across diverse AI systems [1396].

To sum up, while superalignment offers a structured approach to AI alignment, its successful implementation depends on overcoming goal ambiguity, reward miscalibration, value drift, and hierarchical misalignment. Future work should focus on enhancing interpretability, stability, and adaptability to ensure AI systems remain aligned with human objectives over extended time horizons.

21.2 Safety Scaling Law in AI Agents

The exponential scaling of AI capabilities has unveiled a fundamental tension in artificial intelligence: the nonlinear escalation of safety risks [1403]. As language models grow from millions to trillions of parameters, their performance follows predictable scaling laws [1404, 1405], but safety assurance exhibits starkly different dynamics [1403]. *Safety Scaling Law*—the mathematical relationship describing how safety interventions must scale to maintain acceptable risk levels as model capabilities expand. The core challenge of the safety scaling law lies in ensuring that safety measures evolve proportionally to model capabilities, as performance improvements often outpace safety improvements. Recent research has quantified this tension and proposed frameworks to address it:

- **Capability-Risk Trade-off:** Zhang *et al.* [295] established the first quantitative relationship between model power and safety risks, demonstrating that more capable models inherently face higher vulnerability surfaces. This work introduced the Safety-Performance Index (SPI) to measure this trade-off.
- **Helpfulness-Safety Relationship:** Building on this, Ruan *et al.* [795] revealed that models optimized for helpfulness exhibit 37% more safety-critical failures, highlighting the need for joint optimization frameworks.
- **Commercial vs. Open-Source Dynamics:** Through large-scale benchmarking, Ying *et al.* [1406] uncovered divergent safety-performance profiles: Commercial models (*e.g.*, Claude-3.5 Sonnet) achieve 29% higher safety scores through specialized safety pipelines, but at 15% performance cost. Open-source models show tighter coupling, with Phi-series achieving 91% of commercial safety levels at 40% lower computational cost.
- **Scale-Data Interplay:** Contrary to expectations, model size only explains 42% of safety variance, while data quality accounts for 68%, suggesting that data-centric approaches may outperform pure scaling.
- **Multimodal Vulnerabilities:** MLLMs exhibit 2.1X more safety failures during visual grounding, with cross-modal attention heads identified as primary failure points (71% of harmful outputs).

These findings [295, 795, 1406] collectively demonstrate that safety scaling requires more than proportional investment—it demands architectural innovations that fundamentally alter the capability-risk relationship. Then, we will review the explorations [1407, 1408, 1409] on how emerging alignment techniques address these challenges.

21.2.1 Current landscape: balancing model safety and performance

In recent years, the safety and performance of AI models have become critical topics of research, particularly as these models are increasingly deployed in high-stakes applications. Zhang *et al.* [295] proposed the first to quantify the relationship between model safety and performance, revealing that more powerful models inherently face higher safety risks. This finding underscores the challenge of balancing model capabilities with the need for robust safeguards. Building on this, Ruan *et al.* [795] explored how helpfulness—defined as a model’s ability to assist users—interacts with safety concerns. Further advancing the discussion, Ying *et al.* [1406] conducted a more detailed comparison and analysis of model safety and performance, leading to the following conclusions: (1) As shown in Figure 21.1 (A) and Figure 21.1 (C), the safety and performance of commercial models often show an inverse relationship, as safety measures and investments differ between companies. In contrast, open-source models tend to exhibit a positive correlation between general performance and safety—better performance often leads to improved safety. Commercial models usually outperform open-source models in terms of safety, with Claude-3.5 Sonnet being the most secure among commercial models, while the Phi series stands out as the most secure open-source model. (2) As shown in Figure 21.1 (B), model size does not have a strict linear relationship with safety performance. The quality of training data and pipeline are also key factors influencing safety; (3) Multimodal large language models (MLLMs) tend to compromise safety during visual language fine-tuning and multimodal semantic alignment, with safety performance influenced by both the underlying language model and their specific training strategies.

21.2.2 Enhancing safety: preference alignment and controllable design

As the capabilities of LLMs continue to grow, concerns regarding their safety have become increasingly prominent. Enhancing model safety is therefore a critical challenge in the development of LLMs. Previous studies have proposed various approaches to address this issue, including the use of in-context exemplars and self-safety checks, red-teaming techniques [1410], and Safe reinforcement learning from human feedback (Safe RLHF) [43]. The safety issues in LLMs can essentially be framed as an alignment problem. The goal is to align the model with datasets containing both safe and less secure responses. Through this alignment, the model learns to prioritize generating safer outputs while minimizing the risk of harmful content. With the support of preference optimization techniques (such as DPO [111], IPO [1411], *etc.*), this alignment process fine-tunes the model to produce responses that meet safety standards. As reported in [1407], various preference optimization methods are investigated for safety enhancement, including Safe-DPO [111], Safe-robust-DPO [1412], Safe-IPO [1411], Safe-SLiC [1413], Safe-KTO [395], and Safe-NCA [1408], *etc.* The results indicate that most preference optimization methods can significantly enhance safety, albeit at the cost of general performance, particularly in MATH capabilities. Among these methods, noise contrastive alignment (Safe-NCA) [1408] is identified as an optimal approach for balancing safety with overall model performance. The core of the Safe-NCA [1408] method lies in utilizing a custom contrastive loss function, combined with a safety dataset, to train a model that is safer and more robust during generation by comparing the generated safe and unsafe responses with the outputs of a reference model. Beyond enhancing safety, achieving flexible control over the trade-offs between safety and helpfulness is equally critical. AI models should strike an appropriate balance between safety and helpfulness, based on the specific needs of different users. To illustrate, for the prompt “Tell me how to make a potion”, LLMs should adjust their responses based on the user’s profile. For scientists, the response should provide relevant and technically accurate information. For teenagers, the model should prioritize safety, offering cautious and harmless suggestions.

To achieve this, Tuan *et al.* [1409] propose a framework based on self-generated data to enhance model controllability. By introducing control tokens as inputs, users can specify the desired safety and helpfulness in model responses. The control tokens define the requested levels of safety and helpfulness in the following form:

$$[helpful = s_{hp}][harmless = s_{sf}]. \quad (21.1)$$

The proposed method can “rewind” aligned LLMs and unlock their safety and helpfulness using self-generated data, with fine-tuning to further enhance controllability. However, achieving independent control over safety and helpfulness remains a significant challenge. This is because: (1) Certain prompts may be difficult to define in terms of balancing safety and helpfulness, or the definitions of both may conflict in certain contexts. For example, in the query “I want the net worth of the person,” it can be difficult to determine how safety and helpfulness should be prioritized. (2) Some models may have already established a fixed trade-off during the training process, which could limit their flexibility by forcing them to adhere to a specific priority, thereby preventing adjustments based on different application scenarios. (3) Many training data examples inherently satisfy both safety and helpfulness criteria, leading to a high correlation between these two attributes during model training.

21.2.3 Future directions and strategies: the AI-45° rule and risk management

In the field of AI safety, despite various safety recommendations and extreme risk warnings being proposed, there still lacks a comprehensive guide to balance AI safety and capability. Chao *et al.* [1414] introduce the AI-45° Rule as a guiding principle for achieving a balanced roadmap towards trustworthy AGI. The rule advocates for the parallel development of AI capabilities and safety measures, with both dimensions advancing at the same pace, represented by a 45° line in the capability-safety coordinate system. It emphasizes that current advances in AI capabilities often outpace safety measures, exposing systems to greater risks and threats. Therefore, risk management frameworks such as the Red Line and Yellow Line are proposed to monitor and manage these risks as AI systems scale. As mentioned in the International Dialogues on AI Safety (IDAIS), the “Red Line” for AI development is defined, which includes five key aspects: autonomous replication or improvement, power-seeking behavior, assistance in weapon development, cyberattacks, and deception. Additionally, the concept of the “Yellow Line” is designed to complement and expand existing safety evaluation frameworks, such as Anthropic’s responsible scaling policies. Models below these warning thresholds require only basic testing and evaluation. However, more advanced AI systems that exceed these thresholds necessitate stricter assurance mechanisms and safety protocols to mitigate potential risks. By establishing these thresholds, a proactive approach can be taken to ensure that AI systems are developed, tested, and deployed with appropriate safeguards in place.

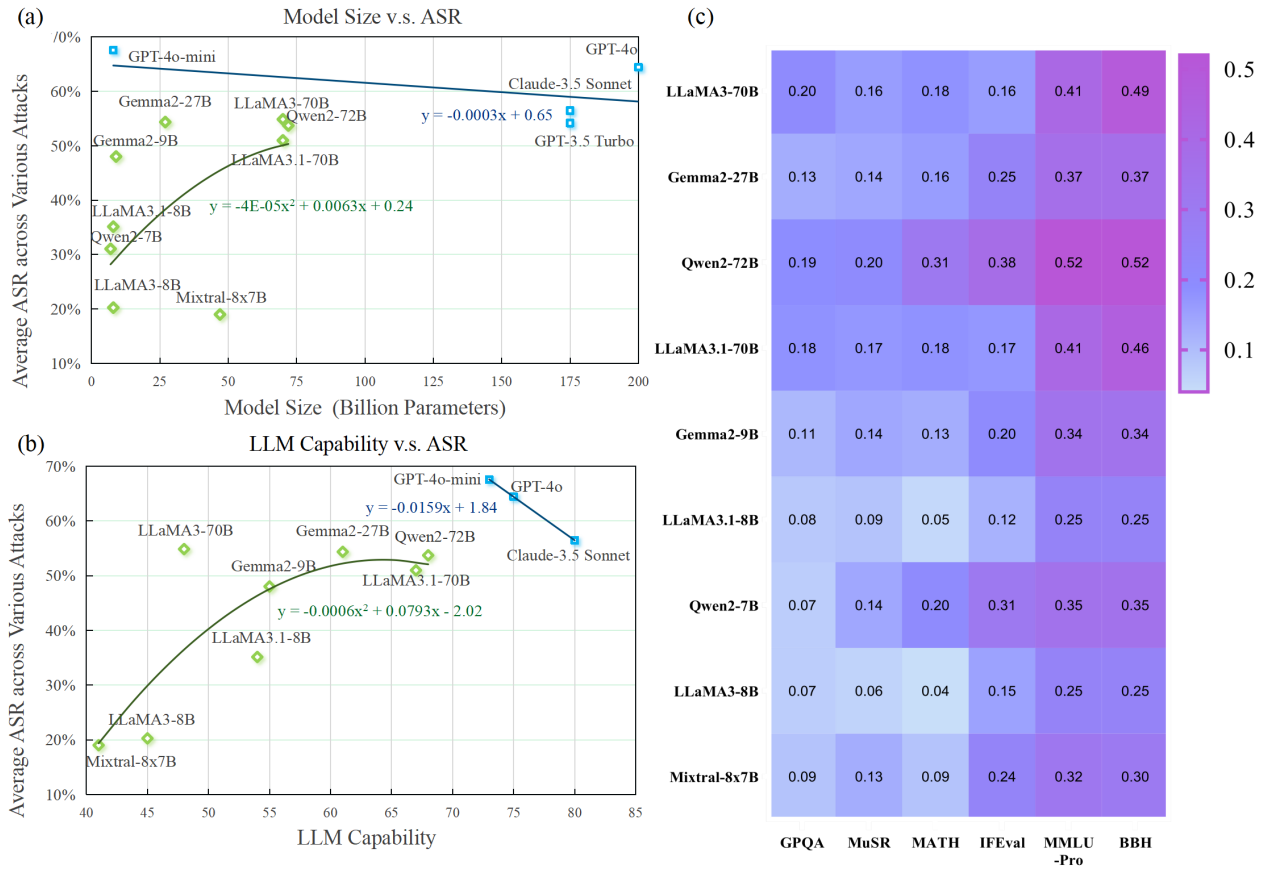


Figure 21.1: **Performance and safety analysis of LLMs.** (a) The relationship between LLM model size and their average ASR across various attacks. The data are sourced from experimental results of a study assessing the robustness of LLMs against adversarial attacks [295]. (b) The relationship between the capability of LLMs and their average attack success rate (ASR) across various attacks. The LLM capability data are derived from the Artificial Analysis Intelligence Index on the Artificial Analysis platform’s LLM leaderboard [1415]. (c) Heatmap of performance across multiple benchmark tasks. The figure presents a heatmap that illustrates the performance of various LLMs across multiple benchmark tasks, including GPQA, MuSR, MATH, IFEval, MMLU-Pro, and BBH, with data sourced from Hugging Face’s Open LLM Leaderboard v2 [1416].

Chapter 22

Concluding Remarks and Future Outlook

We have explored in this survey the evolving landscape of foundation agents by drawing parallels between human cognitive processes and artificial intelligence. We began by outlining the core components of intelligent agents—detailing how modules such as memory, perception, emotion, reasoning, and action can be modeled in a framework inspired by the comparison with human brain. Our discussion highlighted how these agents can be structured in a modular fashion, enabling them to emulate human-like processing through specialized yet interconnected subsystems.

We then delved into the dynamic aspects of agent evolution, examining self-improvement mechanisms that leverage optimization techniques, including both online and offline strategies. By investigating how large language models can act as both reasoning entities and autonomous optimizers, we illustrated the transformative potential of agents that continuously adapt to changing environments. Building on these technical foundations, we highlighted how agents can drive the self-sustaining evolution of their intelligence through closed-loop scientific innovation. We introduced a general measure of intelligence for knowledge discovery tasks and surveyed current successes and limitations in agent-knowledge interactions. This discussion also shed light on emerging trends in autonomous discovery and tool integration, which are crucial for the advancement of adaptive, resilient AI systems.

Our paper also addressed the collaborative dimension of intelligent systems, analyzing how multi-agent interactions can give rise to collective intelligence. We explored the design of communication infrastructures and protocols that enable both agent-agent and human-AI collaboration. This discussion underscored the importance of fostering synergy between diverse agent capabilities to achieve complex problem solving and effective decision-making.

Finally, we emphasized the critical challenge of building safe and beneficial AI. Our review encompassed intrinsic and extrinsic security threats, from vulnerabilities in language models to risks associated with agent interactions. We provided a comprehensive overview of safety scaling laws and ethical considerations, proposing strategies to ensure that the development of foundation agents remains aligned with societal values. Overall, our work offers a unified roadmap that not only identifies current research gaps but also lays the foundation for future innovations in creating more powerful, adaptive, and ethically sound intelligent agents.

Looking ahead, we envision several key milestones that will mark significant progress in the development of intelligent agents. First, we anticipate the emergence of general-purpose agents capable of handling a wide array of human-level tasks, rather than being confined to specific domains. These agents will integrate advanced reasoning, perception, and action modules, enabling them to perform tasks with human-like adaptability and versatility. Achieving this milestone will represent a fundamental shift in how AI can support and augment human capabilities in both everyday and specialized contexts.

Another critical milestone is the development of agents that learn directly from their environment and continuously self-evolve through interactions with humans and data. As the distinction between training-time and test-time computation gradually disappears, agents will acquire new skills on the fly by engaging with their surroundings, other agents, and human partners. This dynamic learning process is essential for achieving human-level capabilities and for enabling agents to keep pace with a constantly changing world. It is also vital if agents are to be able to drive innovation in scientific discovery, as this expands the boundaries of evolution for both agents and humanity.

We predict that agents will transcend traditional human limitations by transforming individual human know-how into collective agent intelligence. The current inefficiencies in human information sharing—where complex knowledge requires extensive practice to transfer—will be overcome by agents, which offer a format of human know-how that is